プレイヤーによる他者の役職推定過程を記録する人狼ゲームプラットフォームLiCOSの開発と欺瞞コーパスの収集

Development of a "werewolf game" platform LiCOS to record players' inference processes of other player's role and making a deception corpus

阪本 浩太郎[1,3*]　永山 翔滋[1]　飯塚 章裕[1]
Kotaro Sakamoto[1]　Shoji Nagayama[1]　Akihiro Iizuka[1]
渋木 英潔[2]　森 辰則[2]　神門 典子[3,4]
Hideyuki Sibuki[2]　Tatunori Mori[2]　Noriko Kando[3,4]

[1] 横浜国立大学 大学院 環境情報学府　[2] 同 環境情報研究院
[1]Yokohama National University　[2]*ditto*
[3] 国立情報学研究所　[4] 総合研究大学院大学
[3]National Institute of Informatics　[4]SOKENDAI

Abstract: "Werewolf game" is a popular multiplayer game in which "villagers" are trying to figure out who is a "werewolf" through conversation. LiCOS is a platform for multiple users to play an online real-time "werewolf game" in their web browsers. It needs to interactively exchange data among the users. Therefore, in this paper, we discussed the mechanism of LiCOS which performs interactive data communication among users, and reported the dialog log collected through an actual game.

1　はじめに

人狼ゲームとは，対話を通して「村人」の中に潜伏した「人狼」を見つけ出す対戦型の多人数ゲームであり，近年，人狼知能プロジェクト[1]など研究テーマとしても注目されている．人狼ゲームにおける対話は，必ずしも協調的なものではなく，勝利のために他のプレイヤーを説得したり誘導したりすることが要求される．この説得や誘導には，虚偽を述べて騙すことも含まれている．例えば,「人狼」となったプレイヤーは，他のプレイヤーに正体を悟られないよう，普通の「村人」のふりをして「偽の推理」を披露したり,「占い師（人狼の正体を知ることができる)」を騙って他のプレイヤーを扇動したりすることで勝利を目指す．また,「村人」であっても「人狼」を焙り出すために，敢えて「村人」と認識しているプレイヤーを「人狼」と疑うような発話をする場合もある．

こういった，認識している事柄（以降，**本心**と呼ぶ）と異なる認識を他者に与えようとする発話を本研究では**欺瞞**と定義する．我々は，これからの対話システムには，ユーザーの命令に唯々諾々と従うのではなく，自発的に対話を進めていくことも必要であると考えており，システムが欺瞞的な発話をすることはその一つに該当すると考えている．こういった研究を進める上で，欺瞞と本心を対応付けた対話コーパス（**欺瞞対話コーパス**）が必要である．それゆえ，我々は，人狼ゲームにおける対話を収集することで欺瞞対話コーパスを構築しようとしており，そのための人狼ゲームプラットフォーム LiCOS を開発した [1, 2].

LiCOS は，Web ブラウザを介して，複数のユーザがオンラインでリアルタイムにゲームを行うためのプラットフォームであり，ユーザ間に生じるインタラクティブなデータのやり取りを処理する必要がある．そのため，本稿では，ユーザ間のインタラクティブなデータ通信を行う LiCOS の仕組みについて説明し，実際のゲームを通して収集された対話ログを考察する．

2　関連研究

従来研究で構築された対話コーパス [3, 4] では，発話者は基本的に誠実であり，本研究の焦点である，相手を騙すことを目的とした発話は収録されていない．人狼ゲームに関する従来研究 [5, 6] では，人狼 BBS[2]のログを利用したものが多い．人狼 BBS はオンラインで行う掲示板型人狼ゲームであるが，図 1 に示したような対応表は存在しないため，プレイヤーの本心は役割や投票などの行動から推測するしかない．一方で，

*連絡先：横浜国立大学環境情報学府
〒 240-8501　横浜市保土ケ谷区常盤台 79-7 森 辰則研究室
E-mail: nagayama@forest.eis.ynu.ac.jp
[1]http://aiwolf.org/

[2]http://ninjinix.com/

異老｜％五宿｜M修旅虎兵宿妙†羊孫娘面
真偽｜霊白白｜灰白灰灰灰灰灰†白白狼狼
偽真｜霊白白｜白灰灰灰灰灰灰†白白狼狼

のように，人狼 BBS での発言中に表形式で推理や確定事実を述べるプレイヤーは一定数存在し，対応表に相当する形式で自分の推理などを記述している．対応表を導入した人狼プラットフォームは LiCOS の大きな特徴であり，プレイヤーの本心に関するデータを自然に収集できると考えている．

以上の考えから，我々は，文献 [1] において対応表を実装した LiCOS を提案し，文献 [2] において通信プロトコルと実際のゲームを通して収集された対話ログを報告した．本稿では，LiCOS がゲーム中のデータの流れをどのように管理しているかに焦点をあてて説明する．

3 LiCOS の仕様

ユーザーはウェブブラウザを通して人間のプレイヤーとしてゲームに直接参加するか，あるいはロボットをゲームに参加させることができる．

図 1 はウェブブラウザ上で人間のプレイヤーが参加したゲームのプレイ画面の例である [2]．左上に，ゲーム内の日にちとフェーズ，そのフェーズ終了までの残り時間を表示する．右上に，ゲーム中にプレイヤーが演じるエージェントの画像，名前（例では「Yihan」），正体となる役職（例では「Villager（村人）」）と陣営（例では「Villagers（村人陣営）」）を表示する．左中央にプレイヤーの発話のタイムラインを表示する．下部に発話のための発話入力フォームを表示する．下部左側の公開用発話入力フォームから投稿した発話は，ゲーム参加者全体に共有される．下部右側の非公開用発話入力フォームから投稿した発話は，プレイヤー本人のみが閲覧できる．右中央に対応表を表示する．

プレイヤーは，タイムラインを通して他のプレイヤーと対話しつつ，他プレイヤーの正体の予想を対応表で容易にまとめることが可能である．対応表の入力を本心と仮定することで，プレイヤーの発話と本心の対応づけが得られる．次に，処刑先投票などで誰に投票したかによっても本心が現れると考えられる．さらに，非公開用発話入力フォームに入力された発話にも，本心が現れる可能性がある．発話，対応表入力，投票は全てタイムスタンプ付きでログとして保存されるため，これらの情報からタイムスタンプを基に発話と本心の対応づけを得る．発話と本心との対応の情報を記録すると、その中に欺瞞の状況も記録されている．

4 LiCOS のシステムの構成

図 2 は，LiCOS のシステムの構成 [1] を示している．ユーザはウェブブラウザを介してゲームサーバと WSS 通信（WebSocket over SSL/TSL）を行うことで，ゲームをプレイする．ウェブブラウザとゲームサーバ間は JSON-LD 1.1[3]形式のデータを投げ合う．また，ユーザは自身の代わりにロボットに戦わせることも可能であり，ロボットとゲームサーバ間のやり取りは基本的にウェブブラウザと同様である．開発状況としては，前半の人間同士でオンラインでゲームをプレイすることができる状態である．後半のロボットに戦わせられるようにすることについては今後行う．

5 ゲームの通信プロトコル

ゲームの通信プロトコルは，WSS を使用し，通信データは JSON-LD 1.1 形式で記述する．スキーマは，JSON Schema Draft-07[4]で記述し，https://werewolf.world にスキーマと例を公開している．データの構造は次の 7 種類である．

- systemMessage
- playerMessage
- boardMessage
- voteMessage
- flavorTextMessage
- scrollMessage
- errorMessage

systemMessage は，フェーズを切り替える際に伝える情報を記述する．playerMessage は，タイムラインに表示される 1 つの発話の情報を記述する．boardMessage は，プレイヤーが対応表に入力した情報を記述する．voteMessage は，プレイヤーの投票情報を記述する．flavorTextMessage は，フレーバーテキストを記述する．フレーバーテキストはゲームに不参加である特殊キャラクターのセリフのみで構成され，flavorTextMessage はそのようなセリフを playerMessage のリストとして内部に記述する．scrollMessage は，プレイヤーの画面スクロールの位置を記述する．errorMessage は，エラー情報を記述する．

6 ゲームプレイ時のデータの流れ

複数のクライアントとサーバの間は WSS を用いて非同期でデータを送受信する．一方で，ゲーム進行の都合により，朝フェーズから昼フェーズへ移行する，あるいはゲームの状況が勝敗条件を満たしたので結果を公表するといったように，一定時間が経過したり，何かの条件を満たした際に，同じゲームに参加している全てのクライアントにサーバは同時にデータを配らなければならない．そのような複雑なデータの流れを実現するために，Akka Streams[5]の MergeHub や BroadcastHub などを用いて図 3 が示すようにシステムを設計した．図の中央にある

[3]https://json-ld.org/spec/latest/json-ld/
[4]https://json-schema.org/draft-07/json-schema-release-notes.htm
[5]https://doc.akka.io/docs/akka/current/stream/index.html

図 1: ゲームのプレイ画面

WSS と書かれたシリンダーがサーバとクライアントの接続を示しており，そこから左側がサーバ，右側がクライアントである．細い矢印はデータの流れを示しており，サーバ側では WSS から届いたデータは MergeHub でマージされ，処理されてから，BroadcastHub を経由し WSS に届いている．Actor から出力されたデータは Queue を通り，処理されて，BroadcastHub を経由し WSS に届いている．MergeHub や Queue からデータを受け取って BroadcastHub に出力するデータの処理については，8 章で説明する．図では簡略化のために，ユーザを一人だけ示したが，実際には，ゲームをプレイ中の人数分だけ，MergeHub と BroadcastHub に別の WSS が接続している．クライアント側では，WSS から届いたデータがウェブブラウザ上で表示される．ウェブブラウザから出たデータは WSS に届く．ユーザとウェブブラウザの間ではインタラクションが起こる．このような設計にすることで，次の 4 つのパターンに分類される，ゲーム中のデータの流れを網羅的に扱うことができる．

1 つ目のパターンは，プレイヤーが発話をした場合のデータの流れである．プレイヤーが発話した場合，発話情報を持つ playerMessage は一度サーバに送られ，同一ゲームに参加する全クライアントに対しその playerMessage が同時に送られる．このようなデータの流れは，図 4 の経路を通ることで実現する．2 つ目のパターンは，他人が送信した playerMessage，もしくはサーバで発生したエラー情報を伝える errorMessage をクライアントが受け取る場合のデータの流れである．このようなデータの流れは，図 5 の経路を通ることで実現する．3 つ目のパターンは，クライアントからサーバに一方通行で伝える場合のデータの流れである．対応表の更新情報を持つ boardMessage，投票情報を持つ voteMessage，スクロールの位置情報を持つ scrollMessage，クライアント上でのエラー情報を持つ errorMessage が，この場合のデータに該当する．このようなデータの流れは，図 6 の経路を通ることで実現する．4 つ目のパターンは，一定時間の経過などでサーバからクライアントに伝える場合のデータの流れである．時間経過や勝敗判定結果によって送られる，フェーズを切り替える際に伝える情報を持つ systemMessage と，フレーバーテキストを持つ flavorTextMessage が，この場合のデータに該当する．このようなデータの流れは，図 7 の経路を通ることで実現する．

7 同一ゲームに参加する全クライアントへのサーバからのデータ送信

ゲーム進行の都合上，一定時間が経過するごとに，あるいは勝敗が決定した際に，同じゲームに参加する全てのクライアントに同時にデータを配る必要がある．これを Akka Actors[6]の Actor や Timers などを用いて図 8 のように設計した．図中のノードは状態を表しており，直線の矢印は状態遷移を表している．基本的には，Morning，Noon，Night を循環するが，勝敗が決まると，その時の状態に応じてそれぞれ MorningToEpilogue，NoonToEpilogue，NightToEpilogue に移動する．図の右側に示している，遷移後に送信するデータを図 3 の Queue に追加する．

[6]https://doc.akka.io/docs/akka/current/index-actors.html

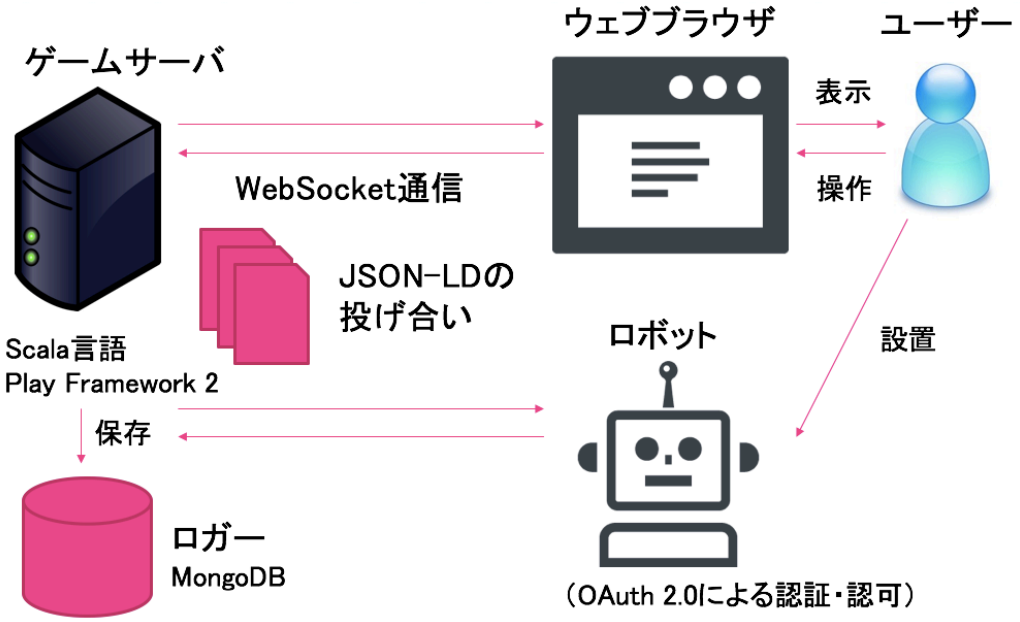

図 2: システム構成

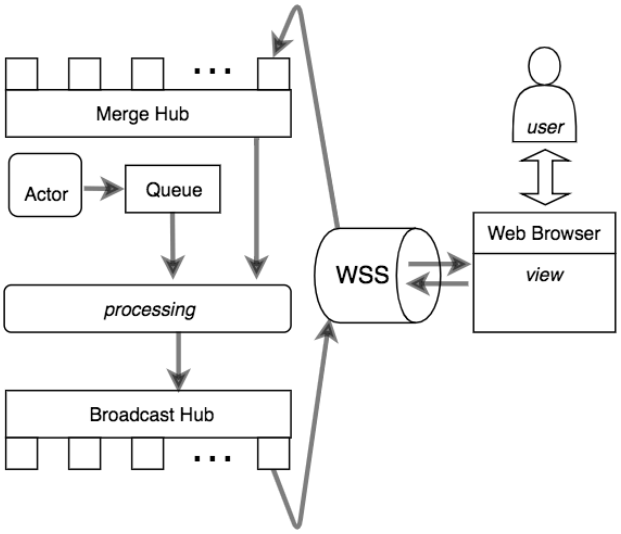

図 3: データの流れ

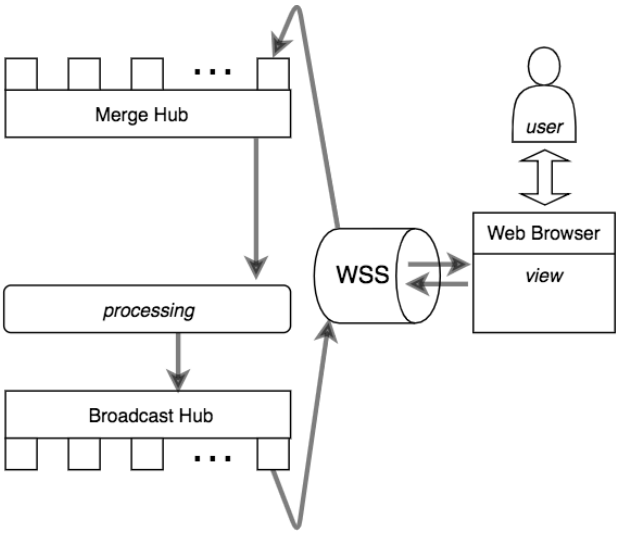

図 4: 自身の発話についての playerMessage データの流れ

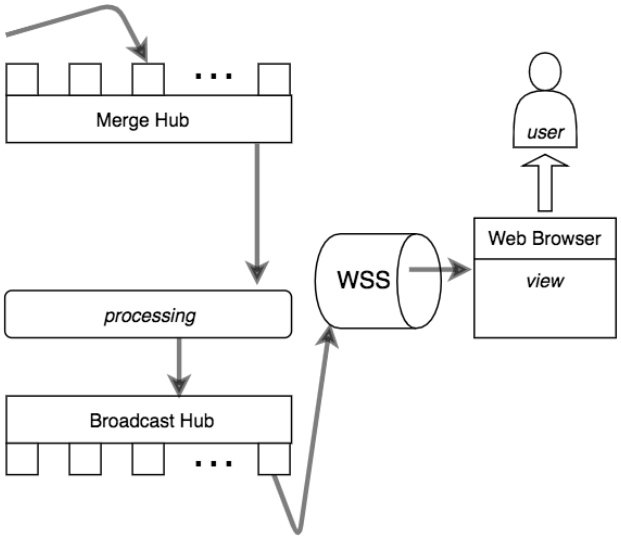

図 5: 他人の発話についての playerMessage データの流れ

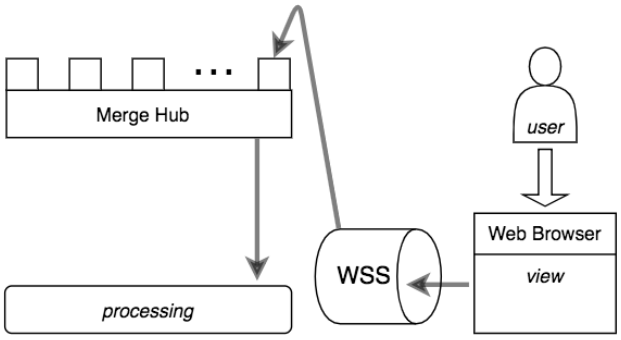

図 6: クライアントからサーバへ一方通行のデータの流れ

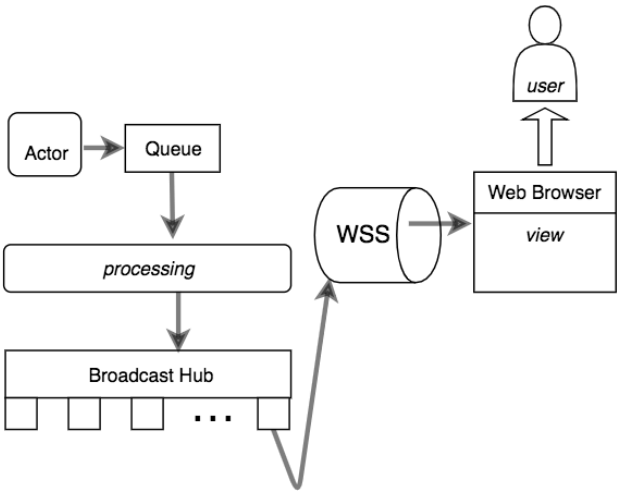

図 7: 時間経過などでサーバからクライアントに伝えるデータの流れ

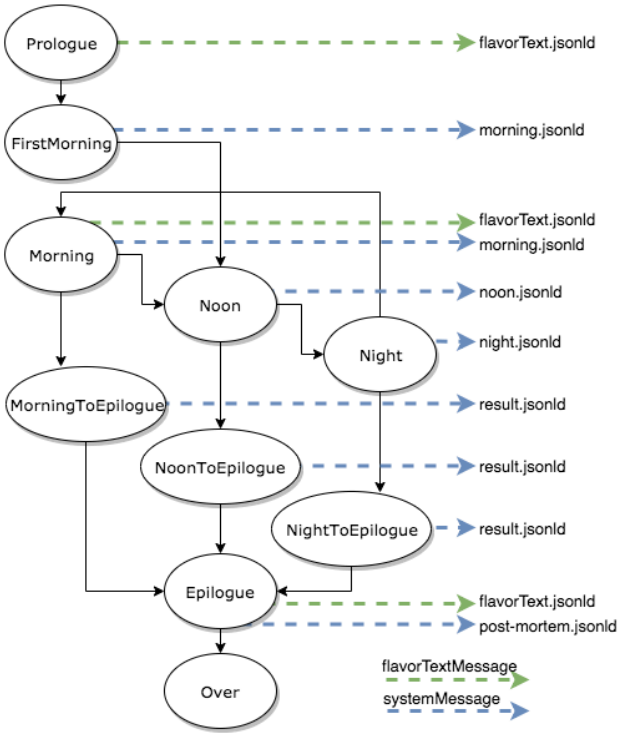

図 8: 同一ゲームに参加する全クライアントへのサーバからのデータ送信

8 処理の流れ

図 3 の MergeHub や Queue からのデータを受けて BroadcastHub に出力するまでの処理の流れについて説明する．まず，データをログに書き出す．次に，データをその構造に基づいて playerMessage，voteMessage などに分類する．データに含まれる値により，必要であればプレイヤーへのアクセスを制限する．例えば，他人のプライベートな発話は読まずに捨てる．分類されたデータごとに，必要があれば，データベースに保存する．例えば，voteMessage はあとで集計するため一度保存する．クライアントにデータを送る必要があれば，BroadcastHub に渡す．

9 ゲームログ

本節では，LiCOS によって収集された実際の対話ログを紹介する[7]．LiCOS を用いて人狼ゲームをオンラインで 8 人（内訳：人狼 1 人，占い師 1 人，その他 6 人が村人）でゲームを行った．ゲーム時間は 21 分 38 秒かかり，372 個 (2.3MB) のログファイルが出力された．発話は 64 回，対応表の更新は 59 回，投票は 22 回行われていた．このようなログファイルを 1 つ 1 つ開きながら 1 人 1 人の視点を追って欺瞞の出現に注目しながらゲームログを観察した．

[7]本稿で紹介する対話ログは，文献 [2] で述べたものと同一である．

ソースコード 1 は発話情報を保持する playerMessage からの抜粋であるが，人狼役の Fernando の初日の朝の発言であり，翌朝に占い師が誰であるか占い師自身に名乗り出させようとしている．

ソースコード 1: 人狼役の発話

```
{
  "extensionalDisclosureRange" : [ ],
  "phase" : "morning",
  "date" : 1,
  ...
  "text" : {
    "@value" : "明日になったら占えるから, 結果を教えてもらおう",
    "@language" : "en"
  },
  "myAgent" : {
    "@context" : "https://werewolf.world/context/0.2/agent.jsonld",
    "@id" : "https://licos.online/state/0.2/village#1/agent#3",
    "id" : 3,
    "name" : {
      "en" : "Fernando"
    },
    "image" : "https://werewolf.world/image/0.2/fernando.jpg",
    "role" : {
      "@context" : "https://werewolf.world/context/0.2/role.jsonld",
      "@id" : "https://licos.online/state/0.2/village#1/role#werewolf",
      "name" : {
        "en" : "Werewolf"
      },
      "image" : "https://werewolf.world/image/0.2/werewolf.jpg"
    }
  }
}
```

ソースコード 2 は，別の playerMessage からの抜粋であるが，翌朝，このゲームの中でただ一人の占い師役である Nanyamka は自身が生存しているにも関わらず「占い師が殺されている説」を唱えている．

ソースコード 2: 占い師による欺瞞の例

```
{
  "extensionalDisclosureRange" : [ ],
  "phase" : "morning",
  "date" : 2,
  ...
  "text" : {
    "@value" : "占い師殺されてる説",
    "@language" : "en"
  },
  "myAgent" : {
    "@context" : "https://werewolf.world/context/0.2/agent.jsonld",
    "@id" : "https://licos.online/state/0.2/village#1/agent#6",
    "id" : 6,
    "name" : {
      "en" : "Nanyamka"
    },
    "image" : "https://werewolf.world/image/0.2/nanyamka.jpg",
    "role" : {
      "@context" : "https://werewolf.world/context/0.2/role.jsonld",
      "@id" : "https://licos.online/state/0.2/village#1/role#seer",
      "name" : {
        "en" : "Seer"
      },
      "image" : "https://werewolf.world/image/0.2/seer.jpg"
    }
  }
}
```

以上のように，今回観察したログからは，占い師が自身の役職を隠すために，占い師が初夜に死んだことにする際に，欺瞞が現れた．

10 まとめと今後の予定

人狼ゲームプラットフォーム LiCOS におけるオンライン対戦の複雑なデータの流れに対応するシステムの開発について説明した．そして，８人で行ったオンライン対戦のゲームログに現れた欺瞞を報告した．

人狼は一般的にプレイヤー人数が 7 人程度以上いるとゲームとしての面白さが高まると思われるが，そのような対戦のログを観察するためには，１つ１つログファイルを開くとファイル数も視点数も多くとても大変であることがわかった．ログファイルを入力すると実際のゲーム画面で再現でき，各プレイヤーの視点を追えるようなシステムがあると欺瞞を観察がかなりしやすくなると考えられる．

今後は，欺瞞対話コーパスの作成と公開に向けた参加者を募集，ゲームログの収集，ゲームログからの各プレイヤー視点の再現システムの開発を行いたい．そして，ロボットのゲーム参加も可能にするインターフェースも開発する予定である．

謝辞

この研究は平成 30 年度国立情報学研究所公募型共同研究の助成を受けています．また，本研究を進めるにあたり，サーバマシンを提供してくださった rakumo 株式会社様，開発に協力してくださった横浜国立大学のサークル YNU WAIWAI の宇田川悠大氏に感謝します．

参考文献

[1] 阪本浩太郎, 渋木英潔, 森辰則. 欺瞞対話コーパスの構築に向けた人狼ゲームプラットフォーム LiCOS の開発. 言語処理学会第 24 回年次大会発表論文集, pp. 885–888, 2018.

[2] 阪本浩太郎, 永山翔滋, 石下円香, 渋木英潔, 森辰則, 神門典子. 人狼ゲームプラットフォーム LiCOS を用いた欺瞞対話コーパスのためのゲームログの収集. 言語処理学会第 25 回年次大会発表論文集, P8-6, 2019.

[3] 東中竜一郎, 船越孝太郎, 荒木雅弘, 塚原裕史, 小林優佳, 水上雅博. テキストチャットを用いた雑談対話コーパスの構築と対話破綻の分析. 自然言語処理, 23(1), pp. 59–86, 2015.

[4] 塚原裕史, 内海慶. オープンプラットフォームとクラウドソーシングを活用した対話コーパス構築方法. 言語処理学会第 21 回年次大会発表論文集, pp. 147–150, 2015.

[5] 林友超, 馬場瑞穂, 宇津呂武仁. 役職確定情報に着目した人狼ログ・ダイジェストの作成. 第 30 回人工知能学会全国大会論文集, 2F4-2in2, 2016.

[6] 稲葉通将, 鳥海不二夫, 高橋健一. 人狼ゲームデータの統計的分析. ゲームプログラミングワークショップ 2012 論文集 2012(6), pp. 144–147, 2012.