2022年度 卒業論文

リアルタイム音声匿名化システム

大阪産業大学 デザイン工学部 情報システム学科
情報教育システム研究室

19H048 塩見仁一朗

リアルタイム音声匿名化システム

19H048 塩見仁一朗

1　はじめに

新型コロナウイルスの影響で大学の授業が対面ではなく、zoom や googleMeet、youtube などを利用してライブ配信で行われるようになった。ライブ配信におけるオンライン授業では録音や録画が容易にでき、授業に出席出来ない生徒のために映像や音声をデータとして残すことが多い。それと同時に現在 Ai を活用した音声合成技術の発達により、音声データを利用して音声元となった人物が実際に話しているかのように声や話し方を再現することが可能になっている。これによって配信の音声や映像を本人の意図しないところで悪用される危険性がある。

2　目的

近年 Ai を活用した音声合成技術の発達が大きく見られる。zoom や googleMeet、youtube などを利用したライブ配信では映像や音声の録音や録画が容易にでき、記録されたデータを音声合成によって本人の意図しないところで悪用される危険性がある。本研究の目的は、話す人物の音声をリアルタイムで別の音声に変換するシステムを開発し、音声を匿名化することによって本人の意図しないところで音声が悪用される危険を防ぐことを目指す。

3　音声匿名化システム

本システムは julius [1] を利用してマイクから入力された音声をテキストに変換し、さらにテキストを OpenJtalk を利用して機械音声に変換する。音声のテキスト化からテキストを機械音声に変換するまでの一連の流れをリアルタイムで行えるようにすることでライブ配信での利用を想定したシステムを実現する。

4　検証

本システムが音声をどれぐらい正確に変換できているのかを検証する。またリアルタイムの音声変換を想定しているため、音声の入力から音声が再生されるまでの時間やテキストから機械音声に変換するまでの時間、機械音声の再生し終わるまでの全体の時間を計測する。

1. 変換した音声の正確性さ
2. マイクの入力から最初の音声の再生までの時間
3. テキストから機械音声に変換するまでの時間
4. 音声が再生し終わるまでの全体の時間

5　結果と考察

本研究の結果、マイクから音声を入力した際にリアルタイムで機械音声に変換することはできるようになった。しかし精度に問題があり、たびたび文章を誤認識することがあった。また一部の単語が完全に読み取ることができなかった。これは juius が内部の辞書を参照して変換を行っており、辞書に単語が存在しないため変換することができなかったと考えられる。

6　まとめ

本研究では話す人物の音声をリアルタイムで別の音声に変換するシステムを開発した。その結果、マイクから入力した音声をリアルタイムで機械音声に変換することができ、目的である音声の匿名化が達成できた。今後の課題として julius の精度の向上や音声のバリエーションがないなどの問題がある。

参考文献

参考文献

[1] 河原達也, 李晃伸. 連続音声認識ソフトウェア julius (特集 研究のツールボックス (2)). 人工知能, Vol. 20, No. 1, pp. 41–49, 2005.

目次

1　はじめに

昨今、新型コロナウイルスの影響で大学の講義が対面授業ではなく Zoom や Google Meet、YouTube などを利用したライブ配信で行われるようになった。ライブ配信における授業では録画や録音が容易にでき、授業に出席できない生徒のために映像や音声をデータとして残すことが多い。しかし現在 artificial intelligence(AI) を活用した音声合成技術の発達により、音声データを利用して音声のもととなった人物が実際に話しているかのうように声や話し方を再現することが可能になってきている。これは素晴らしいことだが悪用も可能であり、データとして残された音声は本人の意思とは関係なく利用される危険性がある。そのために音声を匿名化するシステムが必要であると考える。

第 2 章では本研究の目的について述べる。第 4 章では本研究で開発したシステムの概要を述べる。第 5 章では音声匿名化システムの結果をその考察とともに述べる。第 6 章には研究の成果とともに今後の課題についてまとめる。

2 目的

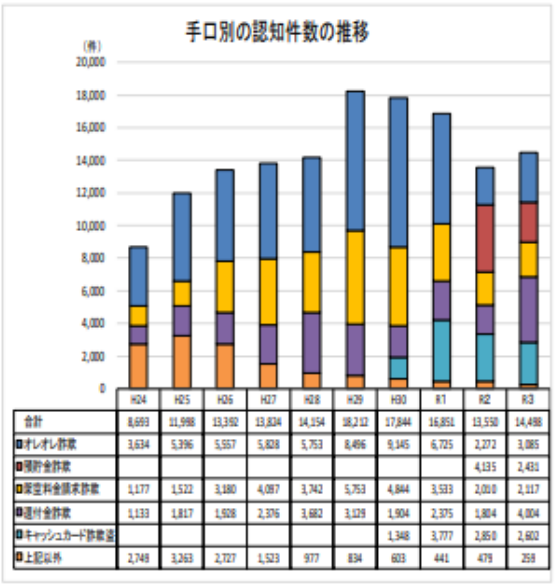
Zoom や Google Meet、YouTube などを介したライブ形式のオンライン授業では話者の音声がデータとして記録され悪用される危険性がある。本研究の目的はリアルタイム音声匿名化システムを実装して、話者がシステムを通して話すことで音声の完全な匿名化ができることを目指すことである。

3　音声データの危険性

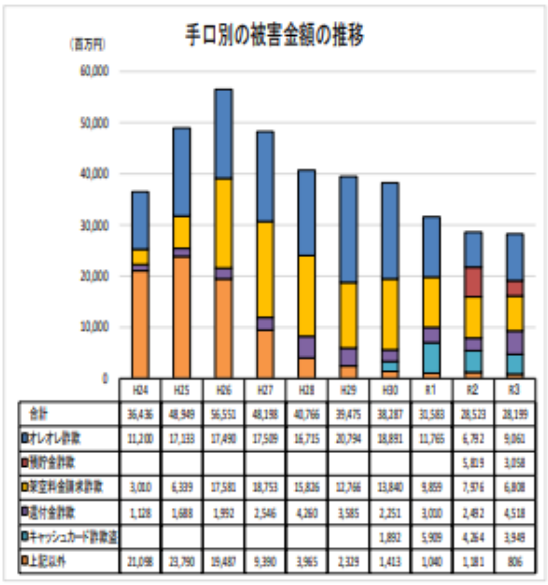
音声データはインターネット上のありとあらゆる場所で利用されており、動画配信や音声配信のオンラインサービスは毎日膨大な量の音声データを生成している。一方で現在 AI 技術の発達が著しくなってり、音声合成によってその人物のクローンを生成することが可能になってきている。実際に現在 SNS 上ではとある実在する人物の声を模した音声データを生成できるサービスが話題になっている。こうした技術は悪用される危険性がある。もしハッカーが保存された音声データにアクセスすることが出来ればその人物になりすますリスクがある。また現在日本では特殊詐欺の被害件数が増加してきており、AI 技術が一般人の手にも届くくらいに広く普及すれば、現在よりも高度な特殊詐欺の増加が予想されるだろう。

3.1　特殊詐欺

警察庁の暴力団対策課と生活安全企画課の報告 [?] によると令和 3 年における特殊詐欺の総認知件数は 14,498 件、被害額は 282.0 億円になり、前年と比べ被害総額は 3.2 億円減少しているもののそう認知件数は 948 件も増加している。中でもオレオレ詐欺の認知件数は 3,085 件で総認知件数の 21.3% を占めており、預貯金詐欺及びキャッシュカード詐欺盗と合わせたオレオレ型特殊詐欺の認知件数は半数以上を占めている。令和 3 年度版の過去年度の認知件数と被害額の推移を図 1 に示す。特殊詐欺において、犯人が電話の相手に対して住所や氏名などの個人情報、現金の保有状況といった情報を探る電話を行うことがある。これは予兆電話といわれ予兆電話でえられた情報をもとに犯行がおこなわれることがある。予兆電話は令和 3 年に 100,515 件、月平均 8,376 件発生しており前年度よりも増加している。予兆電話件数の月ごとの推移を図 2 に示す.。

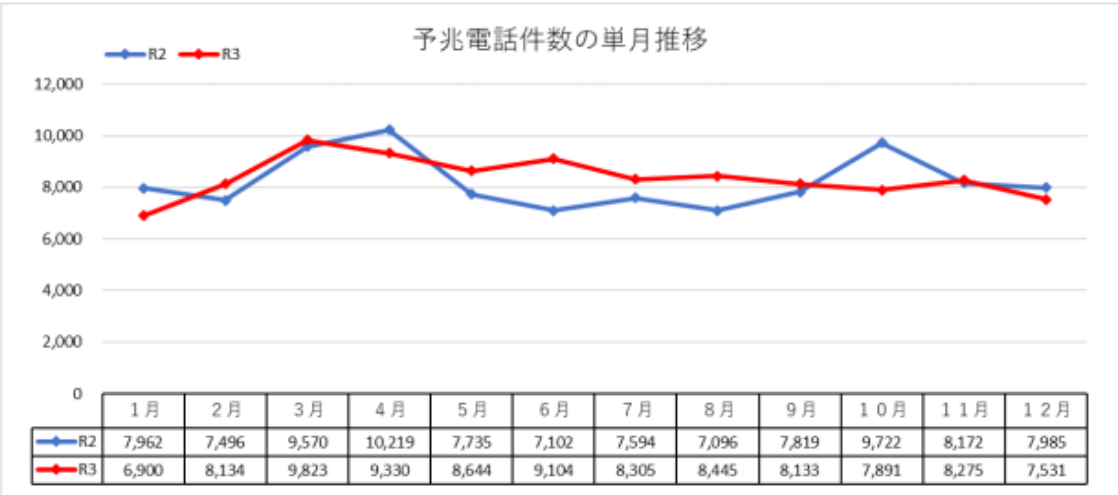


	H24	H25	H26	H27	H28	H29	H30	R1	R2	R3
合計	8,693	11,998	13,392	13,824	14,154	18,212	17,844	16,851	13,550	14,488
オレオレ詐欺	3,634	5,396	5,557	5,828	5,753	8,496	9,145	6,725	2,272	3,085
預貯金詐欺									4,135	2,431
架空料金請求詐欺	1,177	1,522	3,180	4,087	3,742	5,753	4,844	3,533	2,010	2,117
還付金詐欺	1,133	1,817	1,928	2,376	3,682	3,129	1,904	2,375	1,804	4,004
キャッシュカード詐欺盗							1,348	3,777	2,850	2,602
上記以外	2,748	3,263	2,727	1,523	977	834	603	441	479	258



	H24	H25	H26	H27	H28	H29	H30	R1	R2	R3
合計	36,436	48,949	56,551	48,198	40,766	39,475	38,287	31,583	28,523	28,199
オレオレ詐欺	11,200	17,133	17,490	17,509	16,715	20,794	18,891	11,765	6,782	9,061
預貯金詐欺									5,819	3,058
架空料金請求詐欺	3,010	6,339	17,581	18,753	15,826	12,766	13,840	9,859	7,976	6,808
還付金詐欺	1,128	1,688	1,992	2,546	4,260	3,585	2,251	3,010	2,482	4,518
キャッシュカード詐欺盗							1,892	5,909	4,264	3,949
上記以外	21,098	23,790	19,487	9,390	3,965	2,329	1,413	1,040	1,181	806

図１　手口別の認知件数と被害金額の推移 (出典：「令和３年における特殊詐欺の認知・検挙状況等について」(警察庁))



	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月	11月	12月
R2	7,962	7,496	9,570	10,219	7,735	7,102	7,594	7,096	7,819	9,722	8,172	7,985
R3	6,900	8,134	9,823	9,330	8,644	9,104	8,305	8,445	8,133	7,891	8,275	7,531

図２　予兆電話件数の月別推移 (出典：「令和３年における特殊詐欺の認知・検挙状況等について」(警察庁))

4　システムの概要と詳細

音声の匿名化には様々の手法があるが、ここでは音声の完全な匿名化を目指しているため音声変換を利用した方法を考える。本システムは話者が話した際に、入力と受け取った音声を別の音声に変換して出力するシステムである。そのため入力音声を一度テキストデータに変換し、テキストに変換したデータをさらに別の音声に変換する手順を行う。つまり speech-to-text[*1]と text-to-speech[*2]を行う。またライブ配信を想定しているためこれをリアルタイムで変換を行う。本システムの流れを図 3 に示す。

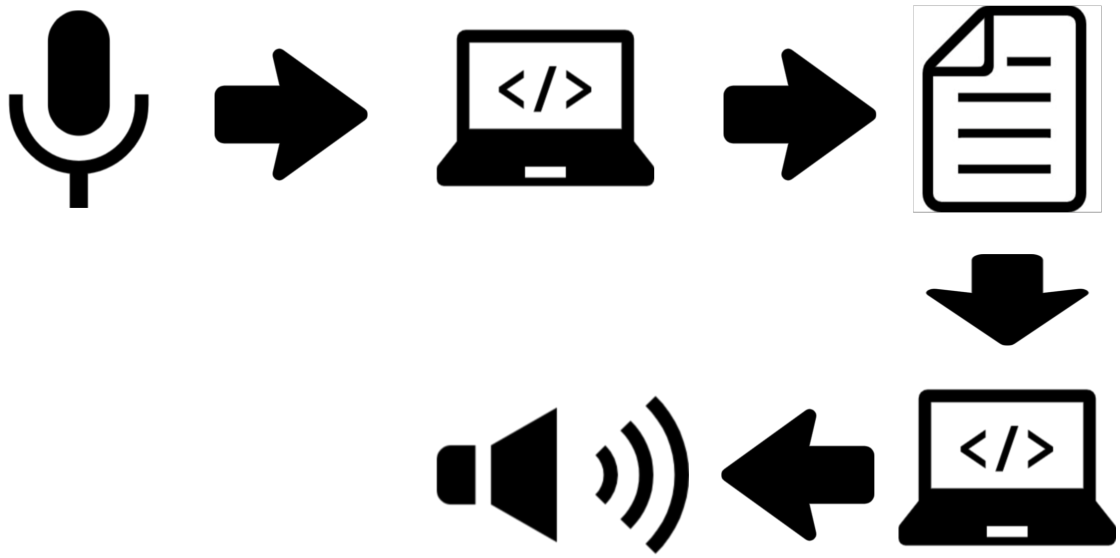


図 3　システムの流れ

4.1　使用するソフトウェア

本システムを実装するにあたり二つのソフトウェアを利用する。

[*1] 音声データをテキストに変換すること
[*2] テキストを音声データに変換すること

1 Julius [**?**]

2 Open Jtalk [**?**]

4.1.1 Julius

Julius とは大語彙連続音声認識を目的として開発されたフリーソフトウェアである。連続的に音声を認識することができ、リアルタイムな処理が可能である。Julius の大きな特徴として汎用性があげられ、ソースコードが公開されており目的に合わせた修正や拡張も可能である。しかし Julius は単語辞書と文法ファイルを利用して音声の認識を行っており、学習機能が実装されていないため新しい単語を認識できるようするためには単語辞書と文法ファイルを直接操作する必要がある。第二の特徴として幅広いプラットフォームに対応していることがあげられ、実際に linux、windows、mac などで動作を確認することができる。利用する際はそのプラットフォームに合わせたコンパイルを行う必要があるが、ディクテーションキットというすでにコンパイルが行われたものを利用することも可能である。このように Julius は汎用性に優れているため、会話ロボットや家電の操作に利用されることもある。本システムでは Julius が speech-to-text の役割を担う。Julius は能動的なシステムであるので、一度実行されれば逐次音声をテキスト化することができ、音声が入力されていない場合は待機状態となる。実際に Julius を実行した際のターミナル上での様子を図 4 に示す。Julius が起動されると<<<please speak>>>と出力され待機状態となる。待機状態にマイクから音声入力を行うと探索が行われ、最終的に sentence1 として結果が出力される。

4.1.2 openJtalk

OpenJtalk とは名古屋工業大学で開発された。日本語向けのテキスト読み上げソフトウェアである。日本語のテキストファイルを入力することで機械音声による読み上げを確認することができる。さらに Open Jtalk もまたフリーソフトウェアであり、誰でも利用することが可能となっている。OpenJtalk は実行された際にテキストファイルの一番最初の行に書かれた文字列を入力として受け取り、wav ファイルとして出力する。本システムでは Open Jtalk が text-to-speech の役割を担う。また Open Jtalk は受動的なシステムであるので、実行を行う際にはトリガーとなるものが必要である。Open Jtalk を実行させた様子を図**??**に示す。実行には「テスト」と書かれたテキストファイルを使用している。上の ls コマンドと下の ls コマンドを見比べてほしい。コマンドを実行した後に test.wa vが生成されているのがわかる。

4.2 システムの動作

本システムは bash を用いて Julius と Open Jtalk の二つのパートに分かれて書かれている。

1 Julus によってマイクから入力された音声をテキストとして出力する。

2 出力されるテキストを stdbuf とリダイレクトを利用してファイルに保存

3 テキストを保存する際に grep、sed を利用して不必要なテキストを排除する

4 inotifywait[*3]を利用してテキストファイルを監視してファイルの更新が行われれば Open Jtalk でテキストを音声ファイルに変換

5 音声ファイルの再生

6 Open Jtalk のパートをバックグラウンドで実行し、フロントで Julius のパートを実行

[*3] inotify-tools をインストールすることで使用できるコマンド。ファイルやディレクトリを監視しイベントの通知をおこなう。

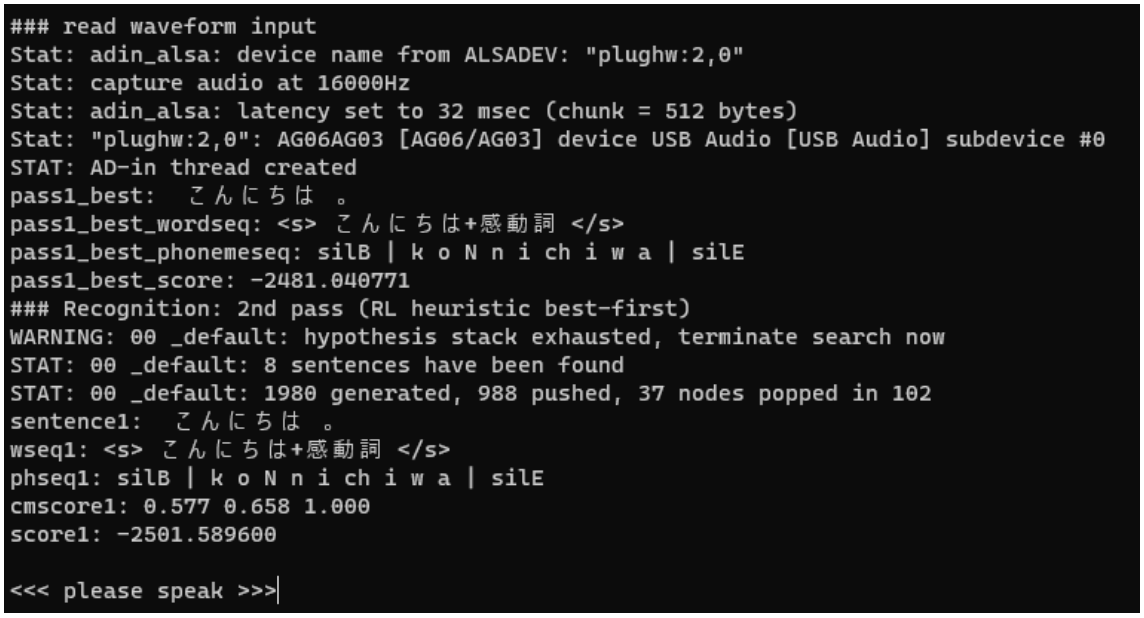


図 4　Julius の実行例：「こんにちは」と音声入力したときの Julius の様子

4.2.1　Julius によってマイクから入力された音声をテキストとして出力する

Julius は起動時に待機状態となり音声が入力された際にテキストに変換を行いコマンドライン上に表示する。また Julius は処理を終えたのちに待機状態へと戻るこれは Julius のそのままの動作である。この様子を図 6 に示す。

4.2.2　stdbuf とリダイレクトを利用してテキストを保存

Open Jtalk につなげるためにテキストの出力先をコマンドライン上ではなくファイルに出力する必要がある。そのため stdbuf コマンドとリダイレクトを利用して出力されるテキストをファイルに保存する（以下このファイルを sample.txt とする）。この様子を図 7 に示す。

図 5　Open Jtalk を実行したときの様子。変換に使用する test.txt には「テスト」と書かれている。

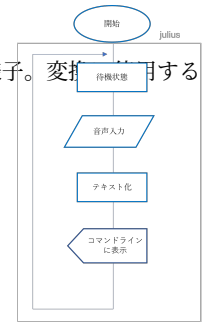


図 6　入力された音声をコマンドライン上にテキストとして表示するフローチャート

4.2.3　テキストから必要な文字列だけを取得する

Julius によって出力されるテキストを保存する際に、不要なテキストも保存されてしまう。そのためリダイレクトでテキストを sample.txt に保存する前にパイプで grep と sed につなげてテキストの処理を行う。この様子を図 8 に示す。

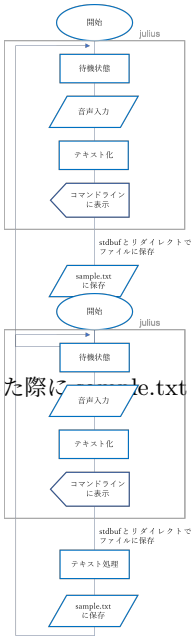


図 7　音声をテキスト化した際に[illegible]e.txt に保存するフローチャート

図 8　テキストの処理を追加した際のフローチャート

4.2.4　ファイルの監視と変換

Open Jtalk でテキストファイルを音声ファイルに変換する。しかし OpenJtalk は受動的なシステムであるので、sample.txt の更新をトリガーとして実行するようにする。その際に inotifywait を利用して sample.txt の監視を行う。この様子を図 9 に示す。

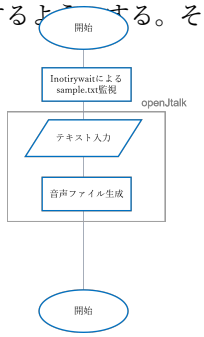


図 9　Open Jtalk を inotifywait を利用して実行するためのフローチャート

4.2.5　音声ファイルの再生

音声ファイルの再生は Open Jtalk のパートで行われる。OpenJtalk で sample.txt を音声ファイルに変換し、その後、apray で音声ファイルの再生を行う。この様子を図 10 に示す。

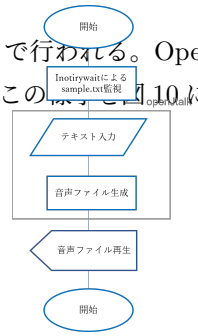


図 10　Open Jtalk を inotifywait を利用して実行し、生成された音声を再生するフローチャート

4.2.6　openJtalk のパートをバックグラウンドで実行

Julius は能動的なシステムであるので、Julius を実行中はほかの処理を実行することができない。そのため Open Jtalk のパートはバックグラウンドで実行する必要がある。バックグラウンドで実行するさいには while

ループを利用して inotifywait で逐次監視を行えるようにする。この様子を図 11 に示す。

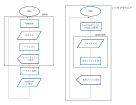


図 11　全体のフローチャート

5　検証

本システムが音声を機械音声に変換した際にどれほど正確に変換できているかを検証する。またリアルタイムの音声変換を目的としているので音声の入力をおこなってから、音声を再生し始めるまでや音声のテキスト化から機械音声に変換するまでのにどれほどの時差が存在するかを計測する。下記に検証内容を示す。

1. 変換したテキストの正確さ
2. マイクの入力から最初の音声の再生までの時間
3. テキストから機械音声に変換するまでの時間
4. 話し始めから最後の音声が再生し終わるまでの全体の時間

5.1　検証方法

文を７つ用意した。これらの文をそれぞれ口頭で読み上げて、変換された音声と元の文がどれほど合致しているかを確認する。時差に関しては script コマンドを使用して音声の入力が行われたとき（音声の入力の際に「s」と手動で入力をおこなう）から変換された音声の再生し終わるまでの時間を表示し計測を行う。また７つの文のうち日常的に使用される文、句読点を含んだ文、Julius の辞書に組み込まれていない可能性がある単語を含んだ文をそれぞれ２つずつ、最後にニュースの原稿に含まれているような専門的な用語を含んだ長文を検証に使用する。

5.1.1　検証する文

1. 日常的に使用される文１
   「私の好きなスポーツはサッカーです。」
2. 日常的に使用される文２
   「今日は晴れのち曇り。」
3. 句読点を含んだ文１
   「これらの文を口頭で読み上げて、変換された音声の正確さを確認する。」
4. 句読点を含んだ文２
   「母の日にはカーネーションを送ります。カーネーションには深い愛という花言葉があるからです。」
5. 和製英語を多く含んだ文１
   「このシステムはユリウスとオープンジェートークを組み合わせて作られました。」
6. 和製英語を多く含んだ文２
   「本日受け取ったデータのソースをいただけますか。」
7. ニュースの原稿に含まれているような専門的な用語を含んだ長文
   「北海道の南東の低気圧は発達しながら北海道付近を進み、その後これらの低気圧が日本の東へ進み、明日の朝にかけて日本付近は強い冬型の気圧配置となる見込みだ。」

5.2　検証結果

5.2.1　日常的に使用される文１

日常的に使用される文１を音声入力した結果を図 12 に示す、「わたしの好きなスポーツは」まではうまく変換できたものの「サッカー」を「作家」と誤変換しているのがわかる。この結果のそれぞれの計測時間を以下に記す。

1. マイクの入力から最初の音声の再生までの時間　3 秒

2. テキストから機械音声に変換するまでの時間　0 秒
3. 話し始めから最後の音声が再生し終わるまでの全体の時間 7 秒

図 12　日常的に使用される文 1「私の好きなスポーツはサッカーです。」を読み上げた結果

5.2.2　日常的に使用される文 2

日常的に使用される文 2 を音声入力した結果を図 13 に示す。問題なく変換することができていることがわかる。この結果のそれぞれの計測時間を以下に記す。

1. マイクの入力から最初の音声の再生までの時間　3 秒
2. テキストから機械音声に変換するまでの時間　1 秒
3. 話し始めから最後の音声が再生し終わるまでの全体の時間 5 秒

図 13　日常的に使用される文２「今日は晴れのち曇り。」を読み上げた結果

5.2.3　句読点を含んだ文１

句読点を含んだ文１を音声入力した結果を図 14 に示す。「これらの文を口頭で読み上げて」の部分の変換には成功しているが、「変換された」が変換されず、「音声」が「温泉」と誤変換されているのがわかる。また息継ぎのタイミングで文が２つに区切られ、変換されているのがわかる。この時のそれぞれの計測時間を以下に記す。

1. マイクの入力から最初の音声の再生までの時間　5 秒
2. １度目にテキストから機械音声に変換するまでの時間　1 秒
3. ２度目にテキストから機械音声に変換するまでの時間　1 秒
4. 話し始めから最後の音声が再生し終わるまでの全体の時間 11 秒

図 14　句読点を含んだ文１「これらの文を口頭で読み上げて、変換された音声の正確さを確認する。」を読み上げた結果

5.2.4　句読点を含んだ文２

句読点を含んだ文２を音声入力した結果を図 15 に示す。「母の日にはカーネーションを送ります」の部分の変換に成功しているが、句読点を含んだ文１と同様に息継ぎのタイミングで文が２つに区切られており、「カーネーションには」の部分が変換されていないことがわかる。この時のそれぞれの計測時間を以下に記す。

1. マイクの入力から最初の音声の再生までの時間　5 秒
2. １度目にテキストから機械音声に変換するまでの時間　1 秒
3. ２度目にテキストから機械音声に変換するまでの時間　1 秒
4. 話し始めから最後の音声が再生し終わるまでの全体の時間 12 秒

図 15　句読点を含んだ文 2「母の日にはカーネーションを送ります。カーネーションには深い愛という花言葉があるからです。」を読み上げた結果

5.2.5　和製英語を多く含んだ文 1

和製英語を多く含んだ文 1 を音声入力した結果を図 16 に示す。「ユリウス」の変換に成功しているが「オープンジェートーク」を「オープンってとこ」と誤変換してしまっていることがわかる。この時のそれぞれの計測時間を以下に記す。

1. マイクの入力から最初の音声の再生までの時間　8 秒
2. テキストから機械音声に変換するまでの時間　1 秒
3. 話し始めから最後の音声が再生し終わるまでの全体の時間 13 秒

図 16　和製英語を多く含んだ文１「このシステムはユリウスとオープンジェートークを組み合わせて作られました。」を読み上げた結果

5.2.6　和製英語を多く含んだ文２

和製英語を多く含んだ文２を音声入力した結果を図 17 に示す。「ソース」の変換に成功しているが「データ」を「データー」と誤変換してしまっていることがわかる。この時のそれぞれの計測時間を以下に記す

1. マイクの入力から最初の音声の再生までの時間　5 秒
2. テキストから機械音声に変換するまでの時間　1 秒
3. 話し始めから最後の音声が再生し終わるまでの全体の時間 9 秒

5.2.7　ニュースの原稿に含まれているような専門的な用語を含んだ長文

ニュースの原稿に含まれているような専門的な用語を含んだ長文を音声入力した結果を図 18 に示す。息継ぎのタイミングで３つに区切られて変換されているのがわかる。「発達しながら」を「果たしながら」と誤変換したり、

図 17　和製英語を多く含んだ文 2「本日受け取ったデータのソースをいただけますか。」を読み上げた結果

「強い冬型の気圧配置となる」を「強い風がその来なさいそうなる」と誤変換してしまっている。また「その後これらの低気圧が日本の東へ進み」の一部が変換されておらず、「わずか二本の東」と誤変換されてしまっているのがわかる。

1. マイクの入力から最初の音声の再生までの時間　7 秒
2. 1 度目のテキストから機械音声に変換するまでの時間　0 秒
3. 2 度目のテキストから機械音声に変換するまでの時間　1 秒
4. 3 度目のテキストから機械音声に変換するまでの時間　2 秒
5. 話し始めから最後の音声が再生し終わるまでの全体の時間 23 秒

図 18　ニュースの原稿に含まれているような専門的な用語を含んだ長文「北海道の南東の低気圧は発達しながら北海道付近を進み、その後これらの低気圧が日本の東へ進み、明日の朝にかけて日本付近は強い冬型の気圧配置となる見込みだ」を読み上げた結果

6 結果と考察

検証の結果、音声のテキスト化の際に精度に大きな問題があるとわかった。「サッカー」を「作家」と変換してしまったり、「音声」を「温泉」と変換してしまったりと、変換の誤りが多く見受けられた。さらに句読点を含んだ文１の「変換された」の部分や句読点を含んだ文２の「カーネーションには」の部分など息継ぎのタイミングのすぐ後ろの言葉が変換されない可能性があることがわかった。また「オープンジェートーク」が「オープンってとこ」と誤変換されているが、これは Julius の単語辞書に「オープンジェートーク」が組み込まれていないためだと考える。

6.1 問題点

6.1.1 検証結果の問題点

1. Julius の辞書に組み込まれていない単語は変換することができない。
   すべての単語が組み込まれているわけではない。とくに和製英語や最近使われだした単語などは認識することができないことが多い。そのため別の単語として認識され、そのまま音声へと変換されてしまう。
2. 単語を本来のものとは別の単語として認識してしまう。または認識されない。
   Julius は単語辞書と文法ファイルを利用して周波数パターンとのマッチングを行っている。そのため話す本人イントネーションによって口にした言葉とは別の言葉として認識されてしまう。またノイズにも弱く、ノイズが多く含まれていると合致したイントネーションで話したとしても、間違った変換をされてしまう。

6.1.2 その他の問題

1. アプリケーションを終了した際にバックグラウンドに inotifywait が残ってしまう
   当アプリケーションでは Julius が音声をテキスト化したと同時にテキストを機械音声に変換するため inotifywait にバックグラウンドでファイルの監視を行わせている。システムの終了には Ctrl+C で行っているため inotifywait が一緒に終了されないため、バックグラウンドで動作し続けることになる。
2. 起動時にマイクの確認・設定をしなければならない
   現在マイクの設定にはアプリケーションの立ち上げと同時に設定されるように「export ALSADEV=”デバイス番号”」と記載しているが raspi の環境においては、raspi の再起動と同時にデバイス番号が変化してしまうときがある。そのためアプリケーションを正常に起動できない時がある。
3. linux 系の環境でしか使用することが出来ない
   プログラムのコードは bash を用いて書かれているため、windows や android の環境で使用することができない。
4. 音声のバリエーションがない
   OpenJtalk を使用しているため男性的もしくは女性的の機械音声しか存在しない

6.2 考えられる改善策

1. Julius の精度の問題を改善する方法として julius の設定を見直すことが考えられる。Julius は内部の単語辞書に収録されていない単語は認識することが出来ないが、辞書に単語を追加することはできる。それによって認識できなかった単語も認識することができるようになると考える。
2. Julius の精度の問題を改善する二つ目の方法について Julius はオープンソースな音声認識ソフトウェアであるため無料で使用することができる。有料になるが google の「speech to text」や Microsoft の「Speech

Services」の使用をすればより高度な音声認識を行うことができ、精度の問題も大きく改善すると考える。

3. linux 系の環境でしか使用することができない問題に関しては、別の言語で書くことによって linux 系以外の環境でも使用できるようになる。そのためには開発を進めていく必要がある。

7　結論

本研究では、音声を機械音声に変換し、リアルタイムで音声を匿名化するシステムの開発を行った。その結果、音声をリアルタイムで機械音声に変換することができるようになり、本研究の目的であるリアルタイムでの音声の匿名化の実現に成功した。現在音声認識の精度や変換先のレパートリーの少なさが問題としてあげられるが、システムの流れは確立されており、今後さらに高度な音声認識ソフトウェアやテキスト読み上げソフトウェアが登場することが期待されるため、その度にアップグレードを重ねていくことでより使いやすいシステムにできることが考えられる。

謝辞

本研究を進めていく上で、担当教員の大垣 斉准教授からご指導およびご協力を頂きました。また、ご協力頂いた情報システム研究室所属の学生および卒業生の方々に深く感謝したします。

付録 A　ソースコード

A.1　Julius と openJtalk を利用して音声を匿名化するプログラム