株式会社 アクエスト
http://www.a-quest.com/

AqKanji2Koe Linux Manual

1. 概要

本文書は、言語処理ライブラリ AqKanji2Koe Linux をアプリケーションに組み込んで使用するためのプログラミングに関しての方法および注意点を示したものです。

AqKanji2Koe は漢字かな混じり文のテキスト情報を AquesTalk(2)用のアクセント付きの音声記号列に変換するライブラリです。
このライブラリと音声合成ライブラリ AquesTalk(2)を使うことにより、動的に変化する様々なテキストからリアルタイムに音声メッセージを生成できるようになります。

特長

・簡単に組み込み可能
　- テキスト文字列を入力すると音声記号列を返す、シンプルな API
・高速な変換処理
　-約 20 万字/秒の高速な変換処理
・高精度な読み・アクセント付与
　-最大 50 万語の単語辞書とアクセントルールにより、正確な読みとアクセントを生成

2. 仕様

AqKanji2Koe

ライブラリ形式	so 形式 共有ライブラリ
対応 OS	Linux 32bit(i386) / 64bit(x86_64)
入力データ形式	漢字かな混じり文テキスト（UTF-8 または UTF-32)
出力データ形式	AquesTalk(2)音声記号列（UTF-8 または UTF-32)
開発言語	言語 C または C++ （I/F は、C関数呼び出し __stdcall)
プログラムサイズ	約 100KByte(32bit)　約 140KByte(64bit)
辞書サイズ	標準：約7MB（約36万語）、スモール：約5MB（約25万語） ラージ：約13MB（約50万語）
処理速度	約 20 万文字/秒
マルチスレッド	対応
メモリ容量	256MB 以上
依存ライブラリ	libgcc_s.so, libc.so

3. ライブラリ配置

AqKanji2Koe Linux のライブラリは共有ライブラリとなっています。リンク時、および実行時に本ライブラリが必要になります。
以下に一例として、/usr/lib に本ライブラリを配置･登録する方法を示します。これによりリンク時および実行時にライブラリにアクセスできるようになります。配置ディレクトリはお使いの環境に応じて変更してください(/usr/lib64 /usr/lib32 など)。また、バージョンによりライブラリのファイル名が実際と異なる場合があります。

以下を su 権限にて行います。

```
# cp libAqKanji2Koe.so.2.0 /usr/lib
# ln -sf /usr/lib/libAqKanji2Koe.so.2.0 /usr/lib/libAqKanji2Koe.so.2
# ln -sf /usr/lib/libAqKanji2Koe.so.2 /usr/lib/libAqKanji2Koe.so
# /sbin/ldconfig -n /usr/lib
```

4. 関数 API

4. 1. AqKanji2Koe.so

AqKanji2Koe_Create

AqKanji2Koe.h

説明	言語処理モジュールのインスタンス生成と内部データの初期化 生成したインスタンスは、使用後、AqKanji2Koe_Release で解放してください。
構文	void * **AqKanji2Koe_Create** (const char **pathDic*, int **pErr*)
引数	
pathDic	辞書のディレクトリを指定。通常、"<app dir>/aq_dic"。指定した内容は内部で保存される。
pErr	*エラー時にはエラーコードが入る 正常終了時は不定値*
戻り値	インスタンスハンドル エラーの時は0が返る。このとき pErr にエラーコードが設定される。

AqKanji2Koe_Create_Ptr

AqKanji2Koe.h

説明	言語処理モジュールのインスタンス生成と内部データの初期化 生成したインスタンスは、使用後、AqKanji2Koe_Release で解放してください。
構文	void * **AqKanji2Koe_Create_Ptr** (const void **pSysDic*, const void **pUserDic*, int **pErr*)
引数	
pSysDic	システム辞書(通常 aqdic.bin) をメモリ上に読み込んだ先頭アドレスを指定
pUserDic	ユーザ辞書(通常 aq_user.dic) をメモリ上に読み込んだ先頭アドレスを指定 ユーザ辞書を使用しない場合は NULL を指定する
pErr	*エラー時にはエラーコードが入る 正常終了時は不定値*
戻り値	インスタンスハンドル エラーの時は0が返る。このとき pErr にエラーコードが設定される。

AqKanji2Koe_Release

AqKanji2Koe.h

説明	言語処理モジュールのインスタンスを開放
構文	void **AqKanji2Koe_Release** (void * *hAqKanji2Koe*)
引数	
hAqKanji2Koe	AqKanji2Koe_Create の戻り値を指定します。
戻り値	なし

AqKanji2Koe_Convert

AqKanji2Koe.h

説明	漢字かな混じりのテキストを音声記号列に変換(UTF8)
構文	int **AqKanji2Koe_Convert** (void * *hAqKanji2Koe*, const char **kanji*, char **koe*, int *nBufKoe*)
引数	
hAqKanji2Koe	AqKanji2Koe_Create の戻り値を指定します。

kanji	入力漢字かな混じり文テキスト文字列（UTF8, BOM 無し,NULL 終端)
koe	出力バッファ。音声記号列文字列が返る（UTF8, BOM 無し,NULL 終端)
nBufKoe	koe バッファのサイズ[byte] 256 以上を指定。バッファサイズ以上の音声記号列は切り捨てられますので入力テキストの数倍のサイズを指定することを推薦。
戻り値	0:正常終了 それ以外：エラーコード

AqKanji2Koe_ConvertW

AqKanji2Koe.h

説明	漢字かな混じりのテキストを音声記号列に変換(ワイド文字（Unicode)版）
構文	int **AqKanji2Koe_Convert** (void * *hAqKanji2Koe*, const wchar_t **kanji*, wchar_t **koe*, int *nBufKoe*)
引数	
hAqKanji2Koe	AqKanji2Koe_Create の戻り値を指定します。
kanji	入力漢字かな混じり文テキスト文字列（Unicode(UTF-32), NULL 終端)
koe	出力バッファ。音声記号列文字列が返る（Unicode(UTF-32), NULL 終端)
nBufKoe	koe バッファのサイズ[byte] 256 以上を指定。バッファサイズ以上の音声記号列は切り捨てられますので入力テキストの数倍のサイズを指定することを推薦。
戻り値	0:正常終了 それ以外：エラーコード

5. サンプルコード

次に示すコードは、日本語のテキストを標準入力から読み込んで、音声記号列を標準出力に出力するだけの単純なプログラムです(SDK パッケージ内の/samples/Kanji2KoeCmd.cpp と基本的に同じものです)。
基本的な処理の流れは、最初に言語処理部のインスタンスを生成し（25 行目）、1行毎にテキストを読み込んで音声記号列に変換して出力しています。最後の行を処理したら AqKanji2Koe_Release()でインスタンスを開放しています。
このプログラムで出力した音声記号列を AquesTalk(2)に与えることで音声データに変換することができます。

```
#include <stdio.h>
#include <AqKanji2Koe.h>
#include <stdlib.h>
#include <string.h>

#define     NSTR  4096

char * GetPathDic(const char *pathModule);

int main(int ac, char **av)
{
    int iret;
    char kanji[NSTR];
    char koe[NSTR];
    void *hAqKanji2Koe;

    if(ac==1){
        char *pPathDic    = GetPathDic(av[0]);
        hAqKanji2Koe = AqKanji2Koe_Create(pPathDic, &iret);
        free(pPathDic);
    }
    else {
        hAqKanji2Koe = AqKanji2Koe_Create(av[1], &iret);
    }
```

```
    if(hAqKanji2Koe==0){
        fprintf(stderr, "ERR: can not initialize Dictionary(%d)¥n", iret);
        fprintf(stderr, "USAGE: $ AqKanji2KoeCmd (dic_dir) < in.txt > out.koe¥n");
        return iret;
    }

    int i;
    for(i=0; ; i++){
        if(fgets(kanji, NSTR, stdin)==0) break;
        iret = AqKanji2Koe_Convert(hAqKanji2Koe, kanji, koe, NSTR);
        if(iret!=0) {
            fprintf(stderr, "ERR: AqKanji2Koe_Convert()=%d¥n", iret);
            break;
        }
        fprintf(stdout, "%s¥n", koe);
    }


    AqKanji2Koe_Release(hAqKanji2Koe);
    return 0;
}
```

実行時には、AqKanji2Koe.so と辞書データファイルが必要です。
AqKanji2Koe.so は、共有ライブラリの検索パス上に配置されている必要があります。
辞書データファイルは任意の場所に配置できますが、配置したディレクトリのパスを AqKanji2Koe_Create()の呼び出し時に指定する必要があります。

AqKanji2Koe_Create()でインスタンスを生成する場合、辞書データは内部メモリに一旦すべて読み込まれます。一方、AqKanji2Koe_Create_Ptr()では、呼び出し側で辞書データを読み込みます。例えば、サーバ利用などで複数の言語処理プロセスを起動する場合は、メモリマップトファイルを使って複数のプロセスで辞書データのメモリを共有することができます。

AqKanji2Koe_ConvertW()は、入出力がワイド文字になっています。Unicode 環境で開発される場合は、こちらを用いたほうが便利です。
なお、ここで生成した音声記号列を AquesTalk(2)に与える場合は、AquesTalk_Synthe()などの代わりに、AquesTalk_Synthe_Utf16()などのワイド文字版の関数を使用してください。

他の言語（PHP,Rubyなど）での使用方法はここでは示しませんが、呼び出して使用することが出来ると思います。

5.1. サンプルプログラム

コンパイル、リンク

SDK パッケージにはサンプルアプリのソースコードが2つ含まれています。Kanji2KoeCmd.cpp は、上記サンプルと同じもの。Kanji2KoeWChar.cpp は、Unicode 環境でのサンプルプログラムです。以下にコンパイル、リンク方法を示します。

```
$ g++   -I./lib -o Kanji2KoeCmd samples/Kanji2KoeCmd.cpp -lAqKanji2Koe
$ g++   -I./lib -o Kanji2KoeWChar samples/Kanji2KoeWChar.cpp -lAqKanji2Koe
```

-I./lib で、ヘッダファイル AqKanji2Koe.h のパスを指定しています。
-lAqKanji2Koe で、libAqKanji2Koe.so をリンクしています。
上記コマンドで、2つの実行モジュール、Kanji2KoeCmd Kanji2KoeWChar が出来るはずです。

実行

実行時には、辞書データを指定された場所に配置しておく必要があります。Kanji2KoeCmd サンプルプログラムでは実行モジュールと同じディレクトリに辞書ファイルのフォルダ(aq_dic)を配置してください。このときのファイルの配置は以下のとおりです。

```
|- Kanji2KoeCmd
|- Kanji2KoeWChar
|- aq_dic/
```

|- aqdic.bin
|- aq_user.dic (ユーザ辞書は必要に応じて)

端末から ./Kanji2KoeCmdを実行すると、入力待ち状態になります。ここで適当な文を入力すると、音声記号列が出力されます。
Kanji2KoeWChar サンプルプログラムも基本的に同じですが、実行時に引数に辞書データのディレクトリを指定する必要があります。

```
$ echo 音声合成テスト | ./Kanji2KoeCmd
オンセーゴーセーテ’スト。
$ echo 音声記号に変換できます | ./ Kanji2KoeWChar ./aq_dic
オンセイキ’ゴーニ/ヘンカン/デキマ’_ス。
$
```

6. エラーコード表

関数が返すエラーコードの内容は、次の通りです。

値	内容
101	関数呼び出し時の引数が NULL になっている。
105	入力テキストが長すぎる
107	変換できない文字コードが含まれている
200 番台	システム辞書(aqdic.bin)が不正
300 番台	ユーザ辞書(aq_user.dic)が不正
100	その他のエラー

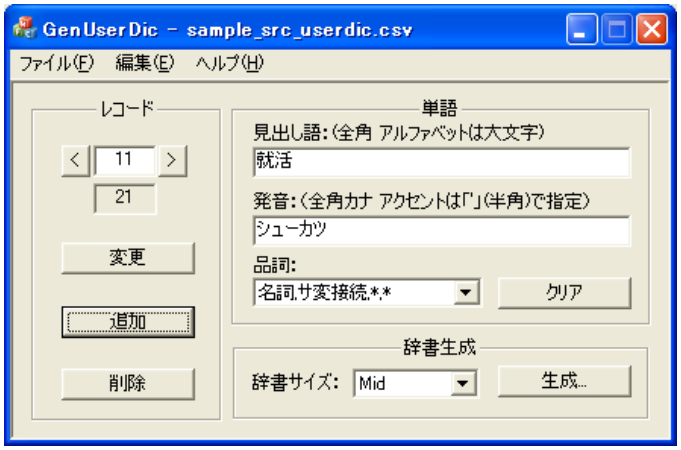
7. 辞書サイズ

この SDK には、サイズの異なる 3 種類のシステム辞書が含まれています。
利用の目的に応じて選択してお使いください。[評価版は標準辞書のみ]

辞書名	フォルダ	サイズ	見出し語数
標準辞書	aq_dic	約7MB	約36万語
ラージ辞書	aq_dic_large	約 13MB	約 50 万語
スモール辞書	aq_dic_small	約 5MB	約 25 万語

8. ユーザ辞書

この SDK には、ライブラリの動作検証用にサンプルのユーザ辞書(aq_user.dic)が含まれています。
独自の単語を追加する場合は、ユーザ辞書作成ツール(GenUserDic.exe)を用います。
[評価版にユーザ辞書作成ツールは含まれていません]
GenUserDic.exe は Windows アプリのため、Windows 環境で動作させる必要があります。



ユーザ辞書の編集

GenUserDic.exe を起動します。
メニュー>ファイル>開く より、付属のサンプルのユーザソース辞書(sample_src_userdic.csv)を開きます。
[見出し語]は、単語をすべて全角で指定してください。またアルファベットは、大文字で記述します。
[発音]部分には、その単語の読みとアクセントを指定します。読みは全角カナで記述し、アクセントは「’」(半角)でアクセント核(声の高さが高→低に変化する部分)に指定します。平板アクセントの場合はアクセントは指定しません。

単語を追加する場合は、[見出し語]、[発音]、[品詞]を指定して、[追加]ボタンを押下します。
表示されているレコードの内容を修正する場合は、[変更]ボタンを押下します。
表示されているレコードを削除する場合は、[削除]ボタンを押下します。

ユーザ辞書ファイルの生成

GenUserDic.exe で内容を変更した後、一旦、メニュー>ファイル>上書き保存で修正内容を保存します。
その後、所望の辞書サイズを選択して、[生成]ボタンを押下し、出力するユーザ辞書ファイル名を指定します。
以上で、ユーザ辞書ファイルが生成されます。
なお、ユーザ辞書のファイル名は aq_user.dic とし、ライブラリの実行時にはシステム辞書(aqdic.bin)と同じディレクトリに配置してください。

コマンドラインによるユーザ辞書ファイルの生成

サンプルのユーザソース辞書(sample_src_userdic.csv)をエディタ等で開くとわかるとおり、ソース辞書は各行が
　見出し語，よみ，品詞
で構成されています。
したがって、エディタ等を用いてユーザソース辞書を直接編集することができます。
注意点として、見出し語,はすべて全角で指定し、アルファベットは大文字で記述。読みは全角カナで記述し、アクセントは「’」(半角)で指定する必要があります。

GenUserDic.exe は、コマンドラインでバッチ的にユーザ辞書を生成することもできます。以下のようにコマンドプロンプト等から実行すると、ユーザソース辞書と同じフォルダにユーザ辞書ファイル(aq_user.dic)が生成されます。

- GenUserDic.exe /MID ユーザソース辞書.csv
- GenUserDic.exe /LARGE ユーザソース辞書.csv
- GenUserDic.exe /SMALL ユーザソース辞書.csv

注意点

単語の読みに、外来語の「テャ」などの一部の音韻は指定できません。指定可能な読みは、下記仕様の「読み記号表」を参照ください。

「音声記号列仕様」
http://www.a-quest.com/download/manual/siyo_onseikigou.pdf

9. 配布パッケージ作成の注意

本ライブラリは、BSD ライセンスに基づいてライセンスされている下記のオープンソースソフトウェアを使用しています。このライセンスの表示は付属の CREDITS ファイルを参照ください。また、本ライブラリを含んだアプリを配布する場合は、CREDITS ファイルまたはその内容が含まれるようにしてください。

- MARISA: Matching Algorithm with Recursively Implemented StorAge
- NAIST Japanese Dictionary : 形態素解析用辞書

10. 履歴

日付	版	変更箇所	更新内容	更新者
2011/01/17	1.0	新規作成		N.Y
2013/06/27	2.0		Ver.2 用に書き換え	N.Y