（株）アクエスト

http://www.a-quest.com/

AqKanji2Koe Mac Manual

1. 概要

本文書は、言語処理ライブラリ AqKanji2Koe Mac 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、Mac OS X 用のライブラリ(Framework)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(Win 版で計測）

・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	Mac framework （ダイナミックライブラリ）
	（リンク時、および実行時、共に本ライブラリが必要になります）
対応 OS	Mac OS X 10.3 以降
CPU	32bit または 64bit (i386/x84_64 Universal Binary)
入力データ形式	漢字かな混じり文テキスト（UTF-8,NULL 終端)
出力データ形式	AquesTalk(2)用音声記号列（UTF-8,NULL 終端)
ビルド環境	Xcode 4.6.3, LLVM 4.2
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）
依存ライブラリ	libstdc++ , libgcc_s, libSystem

3. ファイル

ビルド及び実行時に下記のファイルを使用します。

AqKanji2Koe.framework ： フレームワーク
aq_dic/ ： 辞書ファイル（複数のファイルを含む 実行時に必要)

4. 関数 API

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。

qKanji2Koe_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 漢字かな交じりのテキストを音声記号列に変換

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（UTF-8, NULL 終端)

koe 出力バッファ。音声記号列文字列が返る（UTF-8, 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. 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

6. 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

7. サンプルプログラム

AqKanji2Koe Mac ライブラリパッケージにサンプルプログラムのプロジェクト一式が入っています。
HelloAqKanji2Koe は、任意の漢字かな混じり文を音声記号に変換するアプリケーションです。

7.1. ビルド方法

1. アプリケーションプロジェクトを開く

HelloAqKanji2Koe.xcodeproj をダブルクリックして Xcode でプロジェクトを開きます。

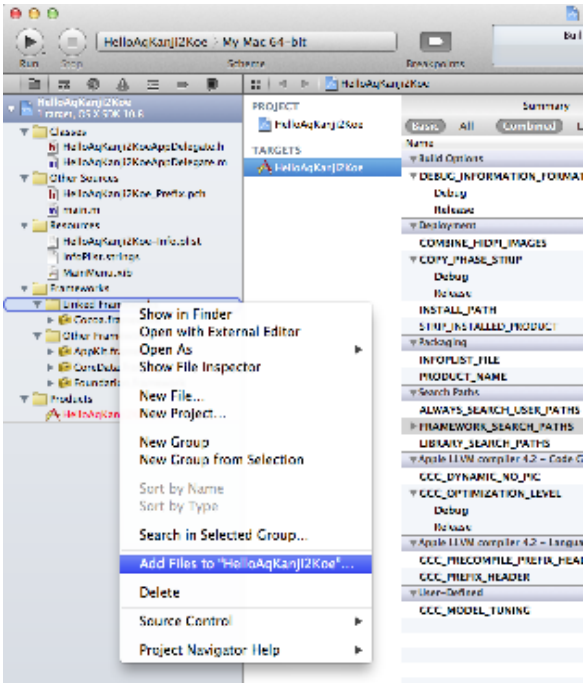
2. プロジェクトにフレームワークの追加]

[評価版では、以下の”AqKanji2Koe.framework”を”AqKanji2KoeEva.framework”に置き換えてください]

パッケージの初期状態では、AqKanji2Koe フレームワークがプロジェクトに含まれていませんので追加します。

「プロジェクトナビゲーター」の [Frameworks]/[Linked Frameworks] の右クリックから [Add Files To “HelloAqKanji2Koe”…]を選択します。

次に、AqKanji2Koe パッケージ内の AqKanji2Koe.framework を選択し、[Add]をクリックします（[Copy items…] のチェックは外したまま）。これで、[Linked Frameworks]に AqKanji2Koe.framework が追加されたのが確認できます。



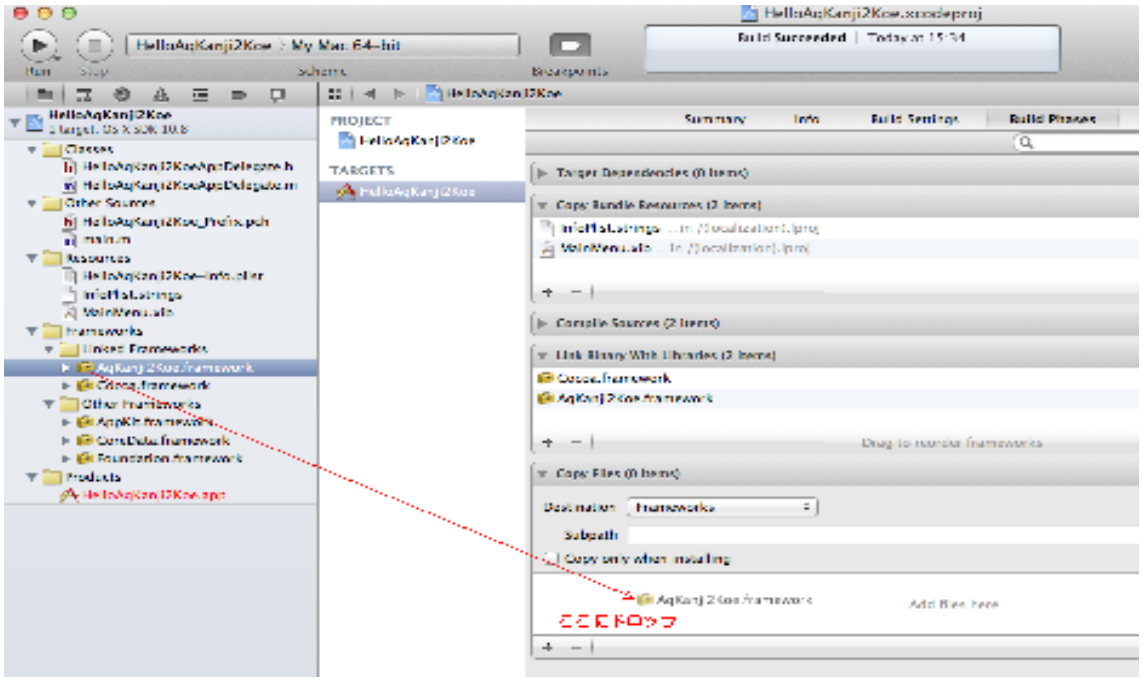
3. ターゲットにもフレームワークを追加

AqKanji2Koe framework は、ダイナミックライブラリなので、実行時にも必要です。

そこで、ビルド後に実行モジュール HelloAqKanji2Koe.app 内に AqKanji2Koe framework がコピーされるように設定します。

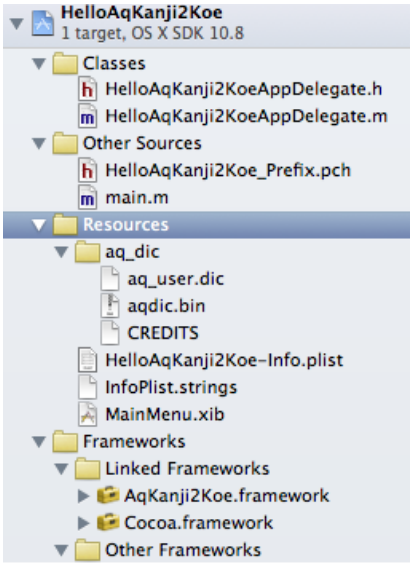
あとから直接フレームワークをコピーしてもよいのですが、ここでは、ビルド時にコピーされる方法を用います。

プロジェクトの Build Phases を開いて、「プロジェクトナビゲーター」の[Frameworks]/[Linked Frameworks]の[AqKanji2Koe.framework]を[Copy Files]の部分にドラッグ &ドロップします。



4. 辞書データの追加

実行時には辞書データも必要です。このサンプルアプリは、実行時にリソースディレクトリにある辞書ファイルを読み込むようにコーディングされています。そこで、SDK 内の辞書データファイル(aq_dic 以下のファイル)を Resources に追加します。
「プロジェクトナビゲーター」の[Resources]の右クリックから[追加]/[既存のフレームワーク]を選択し、SDK の aq_dic フォルダを指定して追加します。



5. ビルド・実行

以上の方法で、フレームワークをプロジェクトに取り込むことができたので、ビルドができます。
ビルドでエラーが無ければ実行してみてください。
テキストボックスに任意の漢字かな混じり文を入力して、Convert ボタンの押下で、下のテキストボックスに音声記号列が表示されれば OK です。[評価版はナ行マ行がすべてヌになります]
実行時に、AqKanji2Koe.framework が見つからないとエラーになる場合は、HelloAqKanji2Koe.app 内に AqKanji2Koe.framework が正しくコピーされていないので、3.に示したとおりに、フレームワークがコピーされるように設定されているか確認します。

7.2. コード説明

次に示すコードは、ボタンが押されたときに呼ばれる関数で、テキストボックスから文字列を取得し、AqKanji2Koe で音声記号列に変換する一連の処理が書かれています。

AqKanji2Koe フレームワークの関数ヘッダをインポートします(10 行目)。
AqKanji2Koe の初期化で辞書ファイルのパスを指定する必要がありますので、バンドルの Resource ディレクトリのパスを取得します(25 行目)。
AqKanji2Koe のインスタンスを生成します(30 行目)。
テキストボックスから漢字を含んだテキストを取得し(33 行目)、UTF-8 の C 言語文字列に変換します(36 行目)。

関数 AqKanji2Koe_Convert()で、音声記号列に変換します。変換結果はkoe配列に戻ります。配列の大きさは変換に十分な大きさを用意してください。このコードではエラー処理を省略していますが、エラー時は関数の戻り値が 0 以外になります。

変換した音声記号列の文字コードは UTF-8 なので、NSString に変換して、テキストボックスへセットします(43 行目)。
最後に、AqKanji2Koe のインスタンスを解放します(47 行目)。

なお、このコードでは、変換の都度 AqKanji2Koe のインスタンスを生成/解放していますが、アプリの起動時に生成し、終了時に解放するようにしたほうが、複数の変換処理を行う場合は、高速に変換することができます。

HelloAqKanji2KoeAppDelegate.m

```
//
//  HelloAqKanji2KoeAppDelegate.m
//  HelloAqKanji2Koe
//
//  Modified by yamazaki on 2013/07/02 for Ver.2
//  Copyright 2013 AQUEST All rights reserved.
//

#import "HelloAqKanji2KoeAppDelegate.h"
#import <AqKanji2Koe/AqKanji2Koe.h>

@implementation HelloAqKanji2KoeAppDelegate

@synthesize window;

- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
    // Insert code here to initialize your application
}


- (IBAction)convKanji2Koe:(id)sender
{
```

```
    // 辞書データのディレクトリパス取得(HelloAqKanji2Koe.app/Resources には辞書ファイル)
    NSBundle *bundle = [NSBundle mainBundle];
25  NSString *path= [bundle resourcePath];
    const char *pathDic = [path UTF8String];

    // AqKAnji2Koe インスタンス生成
    int iErr;
30  void *pAqK2K = AqKanji2Koe_Create(pathDic, &iErr);

    // テキストボックスから文字列取得
33  NSString *strKoe = [textfieldKanji stringValue];

    // NSString 文字列を C 言語文字列(Utf8)に変換
36  const char *kanji = [strKoe UTF8String];

    // 漢字かな交じり文を音声記号列に変換
    char koe[4096];
41  AqKanji2Koe_Convert(pAqK2K, kanji, koe, 4096);

    // C 言語文字列(Utf8)を NSString に変換してテキストボックスへセット
44  [textfieldKoe setStringValue:[NSString stringWithCString:koe encoding:NSUTF8StringEncoding]];

    // AqKAnji2Koe インスタンス解放
47  if(pAqK2K) AqKanji2Koe_Release(pAqK2K);
}
@end
```

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]
恐れ入りますが、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
2013/07/02	2.0		Ver.2 用に書き換え	N.Y