IMPACT External Dictionary interface baseline implementation

Jesse de Does, INL.

For the benefit of IMPACT partners, we provide an implementation of the FineReader SDK interface for external Dictionaries. The main idea of the current setup is that it should be easy to switch between different external dictionary interfaces, and that it would be nice to be able to write the dictionary module in a platform-independent way, independent of FineReader or of any Windows/COM headers and libraries.

The implementation consists of:

1. An executable which is an adapted version of the CommandlineInterface SDK sample program. The executable implements the External Dictionary Interface by proxy: the actual work is done in separate DLL modules which are specified on the command line. The DLL's implement a plain C version of the external Dictionary Interface.
2. The definition a plain C version of the external dictionary interface.
3. An example DLL implementing the plain C version of the external dictionary interface with a static word list with "confidence" information.

The modified CommandLineInterface program

The executable used for the external dictionary demonstrations is an adapted version of the CommandlineInterface FineReader SDK sample program. All documented parameters of this program remain valid. The extra parameters for the invocation of an external dictionary module are:

1. Invoking an externally defined dictionary module

```
CommandLineInterface –erl (--ExternalRecognitionLanguage) <DLL filename> <Dictionary filename>
<Language name>
```

Here *<Language name>* should be one of the language names listed in the FineReader SDK manual.

So, for instance, to perform OCR on the Dutch page1.tif and save the result as RTF, one would have

```
CommandLineInterface –erl BasicExternalDictionaryInterface.dll DutchLexicon.dat Dutch –if
page1.tif –OutputFileFormat RTF –of page1.rtf.
```

As of now, the external language interface cannot be combined with other internal or external languages.

2. An additional output file format option: Page XML.

Plain C version of the external dictionary interface

A "Fuzzy Word" is a representation of a set of word recognition candidates[1] as a list of character recognition variants for each position in a word. For instance, "*d at position 1, o or u at position 2, g at position 3*" is a fuzzy word representing the set *{dog, dug}*. In the FineReader SDK, fuzzy words are represented by a COM Interface. In the plain C version, they are simply arrays of arrays of characters, for instance *{"d", "ou", "g"}* would be the plain C version of the set *{dog, dug}*.

The dictionary DLL should provide entry points for 3 functions:

➔ void * **getDictionary**(`wchar`_t * *filename*)

(returns a handle to a dictionary object. The file referred to may contain some representation of a lexicon, or a configuration file, or whatever other option to construct a lexicon from a string)

➔ int **checkFuzzyPrefix**(`void` * *handle*, `wchar`_t ** *fuzzyWord*, `int` *wordLength*)

Parameters:

Handle	pointer to dictionary object
fuzzyWord	set of recognition candidates specified as an array of character variants for each position in the word
wordLength	length of the fuzzy word

➔ void **checkFuzzyWords**(`void` * *handle*, `wchar`_t *** *FuzzyWords*, `int` * *wordLengthArray*, `int` *nFuzzyWords*, `void` * *callbackOwner*, `void` (**callback)*(`void` *, `wchar`_t *, `int`, `int`))

Parameters:

Handle	pointer to dictionary object
fuzzyWords	array of recognition candidates specified as an array of arrays of character variants for each position in the word.
wordLengthArray	length of the fuzzy words, wordLengthArray[i[is the length of fuzzyWords[i]
nFuzzyWords	number of fuzzy words
callbackOwner	pointer to some object that must be passed as an argument to the callback
Callback	plain C version of the callback to the OCR engine that alerts that a dictionary word has been found among the fuzzy candidates.

[1] Sharing the same character segmentation

To clarify the last two the role of the last two parameters, here is our default implementation of the callback function.

```
void callbackToEngine(void *callbackOwner, wchar_t* w, int confidence, int index)
{
        IExternalDictionaryCallback * callback = (IExternalDictionaryCallback *) callbackOwner;
        fwprintf(stderr, L"Invoke callback with word : %ls  confidence: %d index: %d\n", w,
confidence, index);
        callback -> ExternalDictionaryResult(w, confidence, index);
}
```

We extended the interface with one supplementary method, to enable the dictionary module to suggest a correction candidate. This option is only used with the additional Page XML output format, and its only application is a workaround to deal with the "long s issue", resulting in output like below.

```
<corr sic="Refolutie">Resolutie</corr></Word>
```

➔ `wchar_t * suggestCorrection(void * handle, wchar_t * word)`

Parameters:

Handle	pointer to dictionary object
Word	An arbitrart wide character string

Example invocation of the main DLL function:

```
wchar_t * f1[] = { L"b", L"xoeé", L"kh", L"oaeé", L"oaeéc", L"r" };
wchar_t * f2[] = { L"gbtz", L"eéi", L"sf", L"ct", L"oaéeuc", L"airnu", L"ndt" };
wchar_t **y[] = {f1, f2};
int lengthArray[] = { sizeof(f1) /  sizeof(wchar_t *),
                      sizeof(f2) /  sizeof(wchar_t *) };

(*checkFuzzyWords)(dictionaryHandle, y, lengthArray, 2, NULL,  &myCallback);
```

The baseline DLL implementing static word lists

The baseline DLL provided uses a Double Array Trie to represent the lexicon. Pruning the candidates is a simple recursive trie walk.

Integration with the SDK

We provide an example “CExternalDictionary” implementation of the Finereader IExternalDictionary interface.

The example file LanguageConfiguration.cc shows how to create a TextLanguage object which use the CExternalDictionary class.

RecognizerParamsHolder.cpp contains the function which links the external dictionary to the engine, assuming a IRecognizerParams object. Here is a synopsis:

```
void linkExternalDictionaryToRecognizerParams(CSafePtr<IRecognizerParams> recognizerParams,
wchar_t * languageName, wchar_t * DLLFileName, wchar_t * dictionaryFileName)

{

        CSafePtr<IEngine> engine;

        CSafePtr<ITextLanguage> textLanguage;

        recognizerParams->get_Application( engine.GetBuffer() );


        CExternalDictionary * pExternalDictionary = new CExternalDictionary(DLLFileName,
        dictionaryFileName);



        // function makeTextLanguage… defined in LanguageConfiguration.cc

        makeTextLanguageWithExternalDictionary(pExternalDictionary, textLanguage.GetBuffer(),
        languageName);

        recognizerParams->putref_TextLanguage( textLanguage );

}
```