Word Fragments Based Arabic Language Identification

Hisham El-Shishiny
Emerging Technologies Group
IBM Cairo Technology Development Center
P.O.Box 166, El-Ahram, Giza, Egypt
shishiny@eg.ibm.com

Abstract
We discriminate efficiently between Arabic language and other languages exploiting Arabic script by a word fragments based method. The method makes use of a combination of features characteristic of Arabic language namely function words, prefixes, suffixes and unigrams representing the character set of Arabic script. Results based on 180 samples, selected randomly from the Internet, representing six Arabic based script languages namely Arabic, Persian, Urdu, Pashto, Kurdish and Uighur, achieved 94% recall and 94% precision for Arabic language identification. A key advantage of this approach is that the language model used for identification is transparent and can be tuned and enhanced using linguistic expertise.

1 Introduction
In multi-lingual environments, identifying the language of a piece of text is usually a prerequisite for subsequent processing. In domains with severe constraints on the size of the analyzed texts and on computational resources, language identification of texts still remains an important practical problem.

In this paper we assume that the character encoding scheme of the text is known (in our experiments we used Unicode). We also do not consider the problem of discriminating between Arabic languages and other languages not based on Arabic script as relatively simple.

Word fragments is a new framework which provides a uniform and computationally efficient implementation of the main ideas behind two major language identification schemes – Method of Words and character N-grams. Input text is segmented into tokens by a parser based on Unicode properties of the characters (and exploiting regular expressions mechanism both for text chunking and for detection of non-lexical items like e-mail addresses). Partial morphological analysis of tokens as potential lexical or morphological units is done with the help of dictionaries of words and subwords typical for the languages in question, including word-initial, word-internal and word-final fragments (fragmas for short).

Arabic is a Semitic language with about 221 million speakers in Afghanistan, Algeria, Bahrain, Chad, Cyprus, Djibouti, Egypt, Eritrea, Iran, Iraq, Israel, Jordan, Kenya, Kuwait, Lebanon, Libya, Mali, Mauritania, Morocco, Niger, Oman, Palestinian West Bank & Gaza, Qatar, Saudi Arabia, Somalia, Sudan, Syria, Tajikistan, Tanzania, Tunisia, Turkey, UAE, Uzbekistan and Yemen. Other languages written with the Arabic script are: Arabic, Hausa, Kashmiri, Kazak, Kurdish, Kyrgyz,

2 Related Work
There are in principle two different techniques for the automatic identification of the language of a text document (Grefenstette 1995): the word-based language identification on the one hand and the N-gram based identification on the other.

2.1 Common (short, function) words technique
There is a number of similar methods, based on one proposed by Ingle in 1976 (Ingle 1976). It uses the fact that every language has a set of commonly occurring words. Intuitively, a sentence containing the words and, the, in, would most probably be English, whereas a sentence with the word der would be more likely to be German. One obvious implementation of this technique is to keep a separate lexicon for each possible language, and then to look up every word in the sample text to see in which lexicon it falls. The lexicon that contains the most words from the sample indicates which language was used; weighed sum can be used instead if words are provided with a score.

The main advantage of this method is that words, especially function words (pronouns, prepositions, articles, auxiliaries) tend to be quite distinctive for language identification. However, the literature also attributes the following disadvantages to the words technique (Cavnar and Trenkle, 1994):

• Though common words occur enough in larger texts, they might not occur in a shorter input text.
• Lexicons, especially for highly inflected languages, could be prohibitively big.
• The usage of full form lexicons is hampered by possible misspellings and errors (like those arising from OCR process) and by the presence in texts of out-of-vocabulary words, especially in compounding languages like German.
2.2 N-Gram Technique

The second language modeling technique is based on character N-grams (sequences of N consecutive characters), where N ranges typically from 2 to 5 (Cavnar and Trenkle 1994), (Grefenstette 1995). Similarly to the common words technique, this technique assembles a language model from a corpus of documents in a particular language; the difference being that the model consists of character N-grams instead of complete words.

Absence of linguistic motivation imposes the following disadvantage for N-gram method: N-grams are not as distinctive as function words. For example, the trigrams ‘bo’s, ’bos’, and ‘ost’ are frequently used in the English language, and so the word bost, will have high score to be an English word. However, ‘bost’ is an archaic form not used in the modern English, while bost is an often used abbreviation in Sweden.

3 Arabic Language Identification

3.1 Language Identification Algorithm

The algorithm is based on the detection of word fragments in text. Input text is segmented into tokens by a crosslinguistic parser based on Unicode properties of the characters. Partial morphological analysis of tokens is conducted with the help of finite-state dictionaries of word fragments typical for the languages under consideration and compiled into one finite-state device.

The word fragments are used as features for language identification. Each feature is assigned positive or negative scores (weights) for each language under consideration. The total score for each language is computed based on the individual scores of detected features. The highest score indicates the language of the text.

The system based on the above described algorithm currently recognizes 23 Asian and European languages. Its precision and recall is highly comparable with other known to us technologies, while performance (about 5 Giga Char of text per hour on pentium 4 processors) and memory footprint (1.5 MB) are at least several times better, which is of significant importance for industrial applications.

Following is the description of text parsing for feature detection:

From the initial position in the text, the current character is classified according to Unicode properties. While this character is not alphabetic the position advances to the next character. When an alphabetic is encountered, the text starting at this position is passed to dictionary lookup (the dictionary contains the words, prefixes, suffixes and alphabetic entries). The longest match is returned.

If no match is found, a regular expression lookup is applied to identify the end of this unknown token.

If a match is found, the character after the match is examined. If this character is not a space (or punctuation) then the match context is noted as a prefix match and again regular expression lookup is applied to identify the end of this unknown token.

If, however, this character is a space or punctuation then the match context is set as full word match and scoring commences.

In the case of an unknown token, the token is classified according to the regular expression it matched. If this indicates that the token was word-like, then further feature discovery is possible. In one instance the match context will already be set as prefix match and scoring takes place. The match data found for the entry in the dictionary contains a weight or score to be added to the total score for this language if the context condition of prefix is consistent with the constraints specified in the match data for the entry.

In the case of prefix processing, if scoring is successful, then the score for this entry is stored with its respective part of speech so it can be examined for agreement with that of the following suffix (if found). After prefix processing, the longest suffix is found and the match context is set as suffix match and scoring is processed accordingly, taking the preceding prefix information into account.

The range of the token which is left un-spanned by the prefix and/or suffix matches is then processed character by character to identify alphabetic features. In the dictionary unigrams are identified as distinct from single character prefixes or suffixes by additional constraint information. Therefore the score for a single character entry taken in the context of a prefix/suffix or whole word match will score differently from that same character being taken in the context of an alphabetic feature test and they will not duplicate scoring. Numerals are ignored, since in Arabic script ‘hindi’ numerals or ‘Arabic’ numerals may be used depending on the writing habits in each country.

Processing continues as described until all the text is consumed and the result is a score for each document. Additional processing can be applied to the score such as taking the ratio of score to number of tokens or characters in the text and comparing this to some predetermined threshold. If the score/tokens ratio is lower than the threshold we can reject the document as being Arabic.

3.2 Arabic Language Feature Selection and score assignment

The Arabic language features selected for Arabic language identification are words, word prefixes, word suffixes and unigrams representing the characters used in Arabic script.

3.2.1 Word features

The selected word features for Arabic language are function words which are very distinctive for Arabic language. These function words are prepositions, personal pronouns, demonstrative pronouns, subordinate conjunctions and coordinate conjunctions. In addition some short words very frequently used in Arabic are also selected as word features. A high weight is attributed to the word features.

3.2.2 Prefixes and Suffixes

Arabic is a highly inflected language with complex morphology. Prefixes and suffixes can be attached to words in a concatenative manner, resulting in a single string that represents verb inflections, prepositions, pronouns and connectives. These prefixes and suffixes are characteristics of Arabic language. The length of prefixes and suffixes, in terms of number of characters, increases...
their distinctive power (i.e. the longer the prefix or suffix the more distinctive it is for Arabic language). Therefore, weights are attributed to prefixes and suffixes with values increasing progressively with the length of the prefixes/suffixes.

To provide more compact and efficient model of Arabic language, each Arabic prefix and suffix is provided with information about the part-of-speech of words that accept this affix. Additional bonus score is added, if both Arabic prefix and suffix are found in the same word, and prefix and suffix have compatible part-of-speech information, since short word-initial or word-final fragments, deemed to be Arabic prefixes or suffixes, could also be found in a word belonging to other Arabic script based languages. However, chances that such a word has both prefix and suffix are smaller.

| Prefixes | بين نا تاماه ينهم نمانا |
| Suffixes | ينتمي نمانا نراتب |

Table 1: Sample prefixes and suffixes

<table>
<thead>
<tr>
<th>Feature</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function word</td>
<td>500</td>
</tr>
<tr>
<td>Two character prefix</td>
<td>200</td>
</tr>
<tr>
<td>Three character suffix</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 2: Sample feature weights

3.2.3 Arabic character set

Character set used in Arabic script is in Arabic Range: 0600-06FF in Unicode Standard V4.0, which are base shapes. Extended Arabic letters in the range: 0671 to 06D3 are additional Arabic letters not used by Arabic language but used by the other languages exploiting Arabic script. We have excluded code points 067E, 0686 and 06A4, as they are sometimes used to represent foreign words in Arabic language, from the list of extended Arabic letters, and code points 0671 and 06CC and added to the list of additional Arabic letters not used by Arabic language the following code points: 06D5 06EE 06EF 06FF 06FD 06FE 06FA 06FB 06FC 0671 06CC 067E 0686 06A4.

Characters not used in Arabic language are given high negative scores for Arabic.

4 Results

In this study, we used 180 test files (collected randomly from the Internet), each containing a text pertaining to only one of the following languages that are exploiting Arabic script: Persian, Pashto, Urdu, Kurdish, Uighur and Arabic (see Table 3).

<table>
<thead>
<tr>
<th>Language</th>
<th>Number of files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persian</td>
<td>30</td>
</tr>
<tr>
<td>Pashto</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 3: Language files used in the study

The size of these files was classified into the following ranges (measured by the number of tokens): 6 to 19, 20 to 49, 50 to 99 and 100+. Recall and Precision for Arabic language identification were measured for the word fragment based method (see Table 4).

<table>
<thead>
<tr>
<th>Text size (number of tokens)</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 to 19</td>
<td>85.7%</td>
<td>90%</td>
</tr>
<tr>
<td>20 to 49</td>
<td>100%</td>
<td>94.7%</td>
</tr>
<tr>
<td>50 to 99</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>100+</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4: Recall and Precision for Arabic language

The overall Recall, Precision and Accuracy for Arabic language identification for all test files were found to be:

Recall = 94%
Precision = 94%
Accuracy = 96.6%

5 Conclusion

We have presented in this paper a word-fragment based method that is capable of achieving high discrimination between Arabic language and other languages exploiting Arabic script, along with experimental results.

The method has the key advantages required by industrial applications: high precision/recall, extremely fast processing, small data repository and scalability. The language model used for identification is transparent and can be tuned and enhanced using linguistic expertise.

6 Acknowledgements

We would like to thank John Prager and Thomas Hampp for valuable discussions.

7 References


