High-Speed Identification of Language and Script

Alan Ratner
National Security Agency, 9800 Savage Road, Fort Meade, MD, 20755
asratne@nsa.gov

Ron Loui
Washington University in St. Louis 1 Brookings Drive, St. Louis, MO, 63130
r.p.loui@gmail.com

Abstract

Humans communicate with text in thousands of languages, in dozens of scripts, and a wide variety of binary codes. There is a need to identify the language, script and code of this text to enable follow-on processing such as transcoding, translation, transliteration, routing and prioritization. This paper deals with the implementation of real-time language and script identification on high-speed hardware (principally a ternary content addressable memory) capable of processing network data streams at several gigabits per second.

1. Introduction

We wish to accurately identify the language and script of text messages in real-time on network data streams. Our interest is in those documents generated by humans for humans and not in control and handshaking messages primarily designed for computer-to-computer communications. This restricts the problem to several thousand human languages, each of which may be encoded several different ways for transmission over the network.

The thousands of human languages typically use one of 29 scripts. The scripts consist of phonetic, syllabic or pictorial characters: alphabets (e.g., Latin, Cyrillic, Arabic, Greek, Thai), syllabaries (Korean), and pictograms (e.g., Chinese) [1,2]. Each script may be encoded for transmission or storage in a variety of codes such as Windows Code Pages [3], UTF-8 [4], or HTML special character codes [5]. For example, the English language uses the Latin script to render its characters and this is typically transmitted and stored using US-ASCII (or a functionally equivalent code).

2. Linguistic challenges

There are many difficulties in determining the language of a message. The message may contain only a few words. For example, the message might be simply "Yes". The message might contain several different languages. For example, the message might be "Yes Monsieur Lopez". Much language is international: "White House", "President Bush", "New York City", "Microsoft", "PlayStation" and "Java" do not necessarily imply that the language is English. Many words are used, often with different meanings, in different languages. For example, "son" is a common word in English, French, Spanish and Italian. Many words may be misspelled either intentionally (as in spam) or unintentionally. Language may be transliterated, such as Arabic or Russian appearing in the Latin script. Real people tend to use informal transliteration rules, often using characters that look like rather than sound like their native characters, and show a shocking disregard for formal transliteration rules adopted by governments and academics [6]. The linguistic content is often a minor constituent, typically 1-10%, of the characters within messages. This content is skillfully camouflaged by background noise consisting of headers, footers, formatting, JavaScript [7], etc. Much of this background noise is in English so superficially most messages appear to be in the English language. Finally, real-time processing of packetized messages on simple hardware means that some packets (and hence text) may be missing or appear out of sequence.

3. Technical constraints

The most critical technical constraint is time. At typical network speeds, messages fly by in a few microseconds. The problem can be stated as: "Given a few microseconds, can you figure out the language and script of a message?"

We implemented language and script identification on an existing platform that necessitated performing this process primarily within a Ternary Content Addressable Memory (TCAM) [8]. When queried with a bit pattern, content addressable memory returns the memory location(s) where that pattern appears. This CAM is ternary in that bit patterns may be

0-7695-3019-2/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDMW.2007.117

specified as "0", "1" or "don't care". The application of this TCAM to our high-speed requirement resulted in several constraints. Patterns could be no more than 14 bytes. Each pattern may belong to only one class (i.e., combination of language and script). Patterns may not generate simultaneous matches. (For example, you cannot search for both "men" and "menu".)[1] We can accommodate several hundred patterns per class.

The TCAM interfaces with a Field-Programmable Gate Array (FPGA) and operates on one packet at a time. Packet payloads are typically 1440 bytes. Assuming an average word length of 6 characters plus 1 space character, a packet may contain about 200 words. For each packet, the FPGA accumulates the weighted sum (with each pattern assigned a weight between 1 and 15) of the TCAM pattern matches for each class. The FPGA reports the weighted counts, which saturate at 255, to a CPU that accumulates weighted class counts for each message.

4. Identifying language

The commonly used techniques to identify language use fixed or variable length N-grams [9,10,11,12] or common words [13]. Due to the difficulty of performing quality control on the large corpora used to generate common patterns, we chose to rely primarily on quality control of the common patterns. We selected common words over N-grams to make this quality control process easier. Common words in a specific language can be obtained from published word frequency lists [14,15], word frequency statistics obtained from softcopy corpora, or, for obscure languages for which sufficiently large softcopy corpora are unavailable, by translation of the common words in a similar language.

As word frequency changes with time, published word frequency lists may be out of date. (For example, according to [14], "DOS" is the 450th most commonly used word in English.) Hence we chose to obtain word frequencies from newer corpora.

The rules we set on choosing the patterns to use for each language and scripts were:

1. Eliminate 1 and 2-letter words as being too likely to generate false alarms. Much message content may appear semi-random (such as JPG images) and a 1-character pattern would be expected to generate about 6 matches per packet.
2. Eliminate words commonly used in more than one language.
3. Eliminate words commonly used in container language such as HTML tags, XML tags, JavaScript, and other common formatting.
4. Eliminate international words (e.g., proper nouns).
5. Append a space character before and after each word to minimize false alarms (for languages that normally separate words with spaces). Thus a four-letter word becomes a 6-byte pattern with a smaller false alarm rate at the expense of missing words immediately followed by a punctuation mark. Exceptions to this rule may be made to allow intentional pre-fix or post-fix stemming so that " should" will match " shouldn't" and " shoulder".
6. Make ASCII patterns case insensitive by setting the 6th bit of each character as "don't care". For example, "A" is 0100 0001 while "a" is 0110 0001 in binary so we search for 01X0 0001.
7. Words that normally include diacritics may be written without them. To allow us to distinguish, for example, between normal French and unaccented French, we have segregated words with diacritics into one class and words without diacritics (and possibly words with their diacritics removed) into another class.

Table 1 shows the five most common words in English, French (without diacritics), Spanish (without diacritics), native Arabic, and transliterated Arabic that obey the above rules. Each underscore indicates a single space character.

Table 1. Most common unique words (>2 letters)

English	French	Spanish	Arabic	Arabic Chat
the	_les_	_por_	_الله_	_3ala_
this	_une_	_una_	_على_	_inta_
was	_est_	_como_	_إلى_	_kil_
his	_qui_	_pero_	_عليه_	_wala_
had	_dans_	_porque_	_هذا_	_enta_

Chinese, Japanese and Korean use an extremely large character set and spaces are infrequent. Statistics on Chinese character sequences have been published [16,17]. Table 2 shows the occurrence rates of the most common character sequences within a set of more than 27,000 news articles from 2003-2004 from various Chinese media sources in different countries and regions.

Table 2. Character occurrence in Chinese news corpus

	Character Sequence Length

[1] The prior two constraints provide predictable TCAM response time by preventing more than one pattern match at a time.

	1	2	3	4
Top 100	40.55%	7.99%	1.87%	0.66%
Top 400	73.85%	16.37%	3.99%	1.48%

The table shows that if we search for the 100 most common Chinese 3-characters sequences (typically encoded in 6-bytes) then we would find about 2 pattern matches within a 100-character news-like message; searching for 400 such sequences would yield about 4 pattern matches. Searching for 2-character sequences would quadruple the expected number of pattern matches (at the expense of a greater likelihood of false detections).

5. How many patterns are needed?

Given the practical limitations on the total number of patterns that can be handled by the TCAM, we wish to minimize the number of patterns per class. We can estimate the relative value of the number of patterns using Zipf's law [18] that states that the probability of the n^{th} most common word declines as $1/n^a$ where "a" is 1 or slightly greater. Table 3 shows the relative cumulative likelihood (relative to that for the 100 most common words) of finding the n most common words for a = 1.0 and 1.2. Diminishing returns can be seen after the first few hundred of the most common words; a 16-fold increase in n from 100 to 1600 increases the detection probability by only 23% for a=1.0 or 53% for a=1.2.

Table 3. Cumulative occurrences for n most common words from Zipf's Law (relative to n=100)

n	a = 1.0	a = 1.2
25	0.736	0.826
50	0.867	0.919
100	1.000	1.000
200	1.133	1.071
400	1.267	1.133
800	1.400	1.187
1600	1.534	1.235

An analysis of the nearly 106,000 English language words within a set of 18 short stories [19] was performed. A word was defined as a case-insensitive character string consisting exclusively of letters and the apostrophe. The results in Table 4 show that for the English short stories a value of "a" somewhat less that 1 produces the best agreement. Figure 1 shows a plot of the cumulative probability for words at least 3 characters long (column 4 of Table 4).

Table 4. Cumulative absolute and relative probability for n most common words in English short stories

n	All Words		Words >= 3 Letters	
	Cum. Prob.	Relative to n=100	Cum. Prob.	Relative to n=100
25	34.68%	0.682	28.67%	0.682
50	42.90%	0.844	34.75%	0.827
100	50.84%	1.000	42.04%	1.000
200	58.64%	1.153	50.32%	1.197
400	66.54%	1.309	59.31%	1.411
800	74.21%	1.460	68.41%	1.627
1600	81.77%	1.608	77.56%	1.845

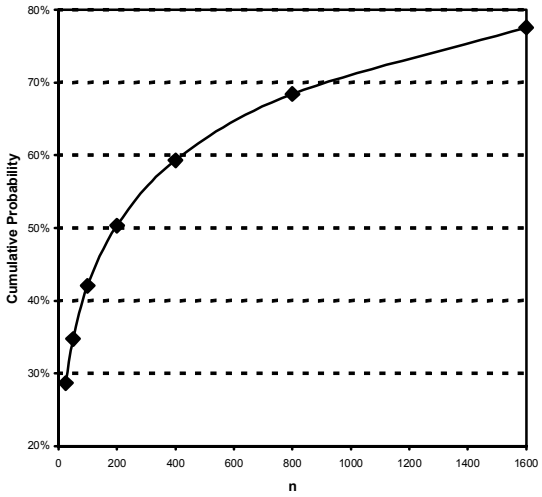

Figure 1. Cumulative Probability of Word Occurrence vs. number of common words in English short stories

word occurrence distribution then we can calculate the probability of identifying the language. If the document contains W words we can compute the probability, P, it contains at least one of the most common words:

$$P = 1-(1-p(n))^W. \quad (1)$$

Here p(n) is the cumulative probability of occurrence of the n most common words. We will assume, from Table 1, that p(n) is 0.5. For short documents of 5, 10 and 15 words this yields probabilities P of 96.9%, 99.9%, and 99.997%, respectively. In realistic documents these probabilities are very much overestimated. The words used in human language are

a prime example of a distribution composed of "Large Numbers of Rare Events" (LNRE) [20]. A prime example would be the 3-word message: "Attack at dawn." According to the word distribution table [14] "attack" is the 1209th most common word in English, "at" is 19th and "dawn" is the 8624th. Unfortunately, our algorithm would ignore the first word as it is not preceded by a space, would ignore the second word as it is less than 3 characters, and would ignore the third word as it is followed by a period rather than a space. It is quite possible to generate documents, known as liponyms, without recourse to common words, much as in the field of lipography where common letters, typically "a" or "e", are avoided.

An example of the pattern matching is provided for the following 115-word paragraph with the most common English words shown in bold. The superscripts indicate whether these words were in the top 50, 100, 200 or 400.

> A Spaniel is a group of gun dog breeds. Spaniels **are**[50] generally smaller dogs with longer coats and drop ears **whose**[200] job is to assist with bird hunting. Spaniels **have**[50] **the**[50] primary purposes of flushing game from dense undergrowth and retrieving game **after**[100] it **has**[50] **been**[50] shot. **Different**[400] breeds reflect **different**[400] emphasis on **the**[50] dogs' uses. At **one**[50] time, spaniels **were**[50] subdivided **into**[50] Land, Field, and **Water**[400] spaniels, according to **the**[50] terrain in **which**[50] **they**[50] worked best. **There**[50] **has**[50] **been**[50] so **much**[100] interbreeding of various gun dogs **over**[100] **the**[50] centuries to achieve additional breeds for new subniches **that**[50] it is **sometimes**[400] difficult to determine **whether**[400] a breed is a spaniel, a retriever, **both**[200] or neither.

Table 5 summarizes the number of unique words and the number of word occurrences in these 4 sets of top words.

Table 5. Pattern matches in 115-word spaniel paragraph

n	Unique Words	Word Occurrences
50	12	17
100	15	20
200	17	22
400	21	27

The 88 words not matched by the top 400 words fell into the following, non-mutually exclusive, categories: 14 were not preceded or followed by spaces, 23 were less than 3 letters long, and the remainder, 52, were examples of "Rare Events" or rejected because they appear in foreign languages or in container language.

Experimentation with a variety of web documents (including Internet forums, legal documents, and Wikipedia pages) revealed that:

- For some languages (e.g., Spanish, native Arabic, native Farsi/Persian) identification was 100% with as few as 50 patterns.
- For some languages, such as Russian, increasing the number of patterns significantly improved performance. This would be expected for languages that make ample use of prefixes, postfixes and conjugations. For these languages performance may be improved with either a greater number of patterns or less generous use of spaces before and after words to permit stem matches.
- For some languages, such as Arabic chat (but not Arabic in native script), increasing the number of patterns significantly improved performance. This is probably due to the wide variety of ways in which people transliterate words. Greater numbers of patterns are required.
- For scripts requiring many bytes per character our system matches short linguistic strings and greater numbers of patterns are required. Arabic script languages are often represented using HTML codes[5] which require 7 bytes each in the ASCII form "&#xxxx;" where xxxx is a four-digit number. Our 14-byte string match is thus limited to sequences of only 2 Arabic script characters encoded using HTML special characters.
- For Chinese (and other Asian languages) the primary difficulty is the lack of spaces parsing character strings into words. We use the most common 3-character Chinese sequences (6 bytes). Due to relative rarity of even the most common 3-character strings, many are required for reasonable recall.

Figure 2 shows the recall of documents from Arabic chat and Russian Internet forums. The Russian recall increased from 80% to nearly 100% as the number of patterns was increased from 50 to 400. The Arabic chat recall was relatively poor; it increased from 43% to only 57% as the number of patterns was similarly increased. It should be noted that the Arabic chat documents were primarily in English with just a few words of Arabic chat mixed in.

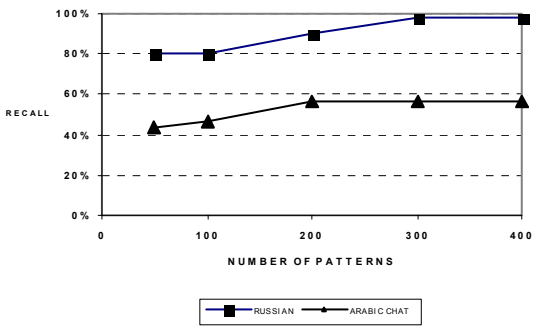

Figure 2. Recall vs. nominal number of patterns

6. Measuring performance

Measuring the performance of language and script identification is complicated by many factors:

- On network links there is a wide variety of formats with varying quantities and varieties of container language.
- Should there be a minimum number of characters or words or sentences of a given language before a message is said to contain that language?
- With multilingual messages, is it important to determine the relative amounts of each language?
- Should a small mistake (e.g., confusing similar languages such as Norwegian with Swedish or Dutch with Afrikaans) count as much as a large mistake (e.g., confusing dissimilar languages such as English with Chinese)?
- How reliable is your confidence in your answer? If your confidence is low, is it better to pass or to provide a potentially erroneous answer?
- Can you distinguish between a language not covered in your table and a non-lingual message?
- Should messages consisting of random words be considered to contain "language"? These messages do not convey meaning but are designed to defeat Bayesian spam filters to deliver a website link.
- Should you include all languages and coding schemes regardless of their *a priori* probabilities? Adding additional classes has complex influences on system performance including:
    - Additional opportunities for false detections due to the increased number of classes.
    - Reduced likelihood of false detections due to newly added classes outscoring random matches within incorrect classes.
    - Reduced pattern matches from words shared with other languages due to their removal from the word lists of other classes. (See Rule 2 from Section 4.)

The best solution to measuring performance is probably a cost matrix with different penalties for different errors. Some credit should be given for identifying the script (e.g., Arabic) and additional credit for identifying the language that is encoded with that script (Arabic, Urdu, Persian, Kurdish, or Pashto). (Similarly, credit should be granted for determining that the script is Cyrillic and for determining that the language is Russian or one of the 60 other languages, from Avar to Yurak, with which it shares the script.)

7. The algorithm

For each text message we identify up to four classes with the highest non-zero numbers of identified common words. We then discard any class whose score failed to exceed its class-specific threshold. Thresholds are typically set to small numbers (such as 1-2) to reduce false alarms. The end result is that each text message is identified as containing 0, 1, 2, 3 or 4 language/script combinations. The case of no reported combinations would be the expected answer for messages without linguistic text or in a language/script not included in the system's linguistic repertoire.

8. Experimental results

Evaluating the correctness of language identification is difficult for the linguistically challenged. We tested using several thousand messages in a variety of *a priori* unknown languages. Our approach was to run the same files through both our system and a more conventional N-gram based program (which was about five orders of magnitude slower). (The N-gram program incorporated a preprocessor to remove container language.) The two agreed on 88% of the messages and for these cases we assumed them to be both correct. This left only a relatively small number of messages in need of evaluation by a human to establish ground truth.

It appears that our system correctly identified the language and script of all messages that contained more than a handful of words in one of the 18 combinations of language and script covered by our pattern set.

Our system was especially effective correctly identifying English language spam containing intentional misspellings (especially with digits and punctuation within words plus use of accented letters). The misspellings typically occurred in the longer, less

common, words. The misspellings created N-grams presumably not observed in the training data.

The principal disagreement between our system and the N-gram program was with spam messages consisting of random words. The N-gram program labeled these as "unknown" while our program labeled these with their specific languages. The correct answer is debatable. Strictly speaking, a language consists of a vocabulary *and* a grammar and this type of spam is grammar-free. Since our N-gram analysis encompassed word-to-word transitions it effectively sensed the grammar while our word-based analysis ignored word sequence information.

9. Discussion

We have demonstrated the ability to accurately perform identification of language and script on a TCAM-based platform by searching for common words within packetized messages at speeds of several gigabits per second. The TCAM supported rates of 10^{13} string compares per second. This may serve to enable such follow-on processing such as transcoding, translation or transliteration. Similar high-speed hardware could be built to perform other knowledge discovery or data mining tasks.

Our system assigns each string pattern an integer weight between 1 and 15. Our efforts to date have used unit weights (i.e., a simple count of pattern matches). Better accuracy might be achieved using weights determined from Term Frequency * Inverse Document Frequency [20, 21] which would assign low weights to common words in common languages (i.e., high document frequency) with higher weights assigned to less common words in less common languages.

Acknowledgments. This research was sponsored, in part, by the Air Force Research Laboratory, Air Force Materiel Command, USAF, under Contract number MDA972-03-9-0001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.

10. References

[1] Nakanishi, A., Writing Systems of the World, Charles E. Tuttle Company, Rutland, VT, (1980)
[2] Katzner, K., The Languages of the World, Routledge, London, (1995)
[3] www.microsoft.com/globaldev/reference
[4] www.unicode.org
[5] agora.org.mk/goodbytes/char/
[6] en.wikipedia.org/wiki/Arabic_Chat_Alphabet
[7] javascript.about.com/library/blreserved.htm
[8] Pagiamtzis, K. and Sheikholeslami, A., Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey, IEEE J. of Solid-State Circuits 41 (2006) 712-727
[9] Beesley, K., Language Identifier: A Computer Program for Automatic Natural-Language Identification of On-line Text, Proc. 29th Annual Conf. American Translators Association (1988) 47-54
[10] Cavnar, W and Trenkle, J.. N-gram based text categorization, Proc. SDAIR-94, 3rd Annual Symp. on Document Analysis and Information Retrieval (1994) 161-175
[11] Kastner, C., Covington, G, Levine, A., Lockwood, J., HAIL: A Hardware-Accelerated Algorithm for Language Identification, 15th Annual Conf. on Field Programmable Logic and Applications (2005)
[12] Kastner, C., HAIL: An Algorithm for the Hardware-Accelerated Identification of Language, M.S. Thesis, Washington University in St. Louis (2006)
[13] Ingle, N., A Language Identification Table, The Incorporated Linguist 15 (1976) 98-101
[14] fr.wiktionary.org/wiki/Wiktionnaire:Listes_de_fr%C3%A9quence
[15] www.artint.ru/projects/frqlist/frqlist-en.asp
[16] Da, J., Reading News For Information: How Much Vocabulary a CFL Learner Should Know, International Interdisciplinary Conference on Hanzi renzhi – How Western Learners Discover the World of Written Chinese, Germersheim, Germany (2005)
[17] lingua.mtsu.edu/chinese-computing/newscorpus/
[18] Zipf, G., Selected Studies of the Principle of Relative Frequency in Language, Harvard University Press (1932)
[19] Jessup, A., ed., *The Best American Humorous Short Stories*, Modern Library, New York (1933) (Project Gutenberg, www.gutenberg.org/files/10947/10947.txt)
[20] Sparck Jones, K., A Statistical Interpretation of Term Specificity and Its Application in Retrieval, J. of Documentation 28 (1972) 11-21
[21] Robertson, S., Understanding Inverse Document Frequency: On Theoretical Arguments for IDF, J. of Documentation 60 (2004) 503-520