Study of Some Distance Measures for Language and Encoding Identification

Anil Kumar Singh
Language Technologies Research Centre
International Institute of Information Technology
Hyderabad, India
anil@research.iiit.net

Abstract

To determine how close two language models (e.g., *n*-grams models) are, we can use several distance measures. If we can represent the models as distributions, then the similarity is basically the similarity of distributions. And a number of measures are based on information theoretic approach. In this paper we present some experiments on using such similarity measures for an old Natural Language Processing (NLP) problem. One of the measures considered is perhaps a novel one, which we have called *mutual cross entropy*. Other measures are either well known or based on well known measures, but the results obtained with them *vis-a-vis* one-another might help in gaining an insight into how similarity measures work in practice.

The first step in processing a text is to identify the language and encoding of its contents. This is a practical problem since for many languages, there are no universally followed text encoding standards. The method we have used in this paper for language and encoding identification uses pruned character *n*-grams, alone as well augmented with word *n*-grams. This method seems to give results comparable to other methods.

1 Introduction

Many kinds of models in NLP can be seen as distributions of a variable. For various NLP problems, we need to calculate the similarity of such models or distributions. One common example of this is the *n*-grams model. We might have several reference data sets and then we may want to find out which of those matches most closely with a test data set. The problem of language and encoding identification can be represented in these terms. One of the most important questions then is which similarity measure to use. We can expect that the performance obtained with the similarity measure will vary with the specific problem and the kind of model used or some other problem specific details. Still, it will be useful to explore how these measures relate to each other.

The measures we are going to focus on in this paper are all very simple ones and they all try to find the similarity of two models or distributions in a (more or less) information theoretic way, except the *out of rank* measure proposed by Cavnar and Trenkle (Cavnar and Trenkle, 1994).

This work had started simply as an effort to build a language and encoding identification tool specifically for South Asian languages. During the course of this work, we experimented with various similarity measures and some of the results we obtained were at least a bit surprising. One of the measures we used was something we have called *mutual cross entropy* and its performance for the current problem was better than other measures.

Before the content of a Web page or of any kind of text can be processed for computation, its language and encoding has to be known. In many cases this language-encoding is not known beforehand and has to be determined automatically. For languages like Hindi, there is no standard encoding followed by everyone. There are many well known web sites using their own proprietary encoding. This is one of the biggest problems in actually using the Web as a multilingual corpus and for enabling a crawler to search the text in languages like Hindi. This means that the content in these languages, limited as it is, is invisible not just to people (which could be just due to lack of display support or unavailability of fonts for a particular encoding) but even to crawlers.

The problem of language identification is similar to some other problems in different fields and the techniques used for one such problem have been found to be effective for other problems too. Some of these problems are text categorization (Cavnar and Trenkle, 1994), cryptanalysis (Beesley, 1988) and even species identification (Dunning, 1994) from genetic sequences. This means that if something works for one of these problems, it is likely to work for these other problems.

It should be noted here that the identification problem here is that of identifying both language and encoding. This is because (especially for South Asian languages) the same encoding can be used for more than one languages (ISCII for all Indian languages which use Brahmi-origin scripts) and one language can have many encodings (ISCII, Unicode, ISFOC, typewriter, phonetic, and many other proprietary encodings for Hindi).

In this paper we describe a method based mainly on character *n*-grams for identifying the language-encoding pair of a text. The method requires some training text for each language-encoding, but this text need not have the same content. A few pages (2500-10000 words) of text in a particular language-encoding is enough. A pruned character based *n*-grams model is created for each language-encoding. A similar model is created for the test data too and is compared to the training models. The best match is found using a similarity measure. A few (5-15) words of test data seems to be enough for identification in most cases.

The method has been evaluated using various similarity measures and for different test sizes. We also consider two cases, one in which the pruned character *n*-grams model is used alone, and the other in which it is augmented with a word *n*-gram model.

2 Previous Work

Language identification was one of the first natural language processing (NLP) problems for which a statistical approach was used.

Ingle (Ingle, 1976) used a list of short words in various languages and matched the words in the test data with this list. Such methods based on lists of words or letters (unique strings) were meant for human translators and couldn't be used directly for automatic language identification. They ignored the text encoding, since they assumed printed text. Even if adapted for automatic identification, they were not very effective or scalable.

However, the earliest approaches used for automatic language identification were based on the above idea and could be called 'translator approaches'. Newman (Newman, 1987), among others, used lists of letters, especially accented letters for various languages and identification was done by matching the letters in the test data to these lists.

Beesley's (Beesley, 1988) automatic language identifier for online texts was based on mathematical language models developed for breaking ciphers. These models basically had characteristic letter sequences and frequencies ('orthographical features') for each language, making them similar to *n*-grams models. The insights on which they are based, as Beesley points out, have been known at least since the time of Ibn ad-Duraihim who lived in the 14th century. Beesley's method needed 6-64 K of training data and 10-12 words of test data. It treats language and encoding pair as one entity.

Adams and Resnik (Adams and Resnik, 1997) describe a client-server system using Dunning's *n*-grams based algorithm (Dunning, 1994) for a variety of tradeoffs available to NLP applications like between the labelling accuracy and the size and completeness of language models. Their system dynamically adds language models. The system uses other tools to identify the text encoding. They use 5-grams with add-*k* smoothing. Training size was 1-50 K and test size above 50 characters. Some pruning is done, like for frequencies up to 3.

Some methods for language identification use techniques similar to *n*-gram based text categorization (Cavnar and Trenkle, 1994) which calculates and compares profiles of *n*-gram frequencies. This is the approach nearest to ours. Such methods differ in the way they calculate the likelihood that the test data matches with one of the profiles. Beesley's method simply uses word-wise probabilities of 'digram' sequences by multiplying the probabilities of sequences in the test string. Others use some distance measure between training and test profiles to find the best match.

Cavnar also mentions that top 300 or so *n*-grams are almost always highly correlated with the language, while the lower ranked *n*-grams give more specific indication about the text, namely the topic. The distance measure used by Cavnar is called 'out-of-rank' measure and it sums up the differences in rankings of the *n*-grams found in the test data as compared to the training data. This is among the measures we have tested.

The language model used by Combrinck and Botha (Combrinck and Botha, 1994) is also based on bigram or trigram frequencies (they call them 'transition vectors'). They select the most distinctive transition vectors by using as measure the ratio of the maximum percentage of occurrences to the total percentage of occurrences of a transition vector. These distinctive vectors then form the model.

Dunning (Dunning, 1994) also used an *n*-grams based method where the model selected is the one which is most likely to have generated the test string. Giguet (Giguet, 1995b; Giguet, 1995a) relied upon grammatically correct words instead of the most common words. He also used the knowledge about the alphabet and the word morphology via *syllabation*. Giguet tried this method for tagging sentences in a document with the language name, i.e., dealing with multilingual documents.

Another method (Stephen, 1993) was based on 'common words' which are characteristic of each language. This methods assumes unique words for each language. One major problem with this method was that the test string might not contain any unique words.

Cavnar's method, combined with some heuristics, was used by Kikui (Kikui, 1996) to identify languages as well as encodings for a multilingual text. He relied on known mappings between languages and encodings and treated East Asian languages differently from West European languages.

Kranig (Muthusamy et al., 1994) and (Simon, 2005) have reviewed and evaluated some of the well known language identification methods. Martins and Silva (Martins and Silva, 2005) describe a method similar to Cavnar's but which uses a different similarity measure proposed by Jiang and Conrath (Jiang and Conrath, 1997). Some heuristics are also employed.

Poutsma's (Poutsma, 2001) method is based on Monte Carlo sampling of *n*-grams from the beginning of the document instead of building a complete model of the whole document. Sibun and Reynar (Sibun and Reynar, 1996) use mutual information statistics or relative entropy, also called Kullback-Leibler distance for language identification. Souter et al.(Souter et al., 1994) compared unique character string, common word and 'trigraph' based approaches and found the last to be the best.

Compression based approaches have also been used for language identification. One example of such an approach is called Prediction by Partial Matching (PPM) proposed by Teahan (Teahan and Harper, 2001). This approach uses cross entropy of the test data with a language model and predicts a character given the context.

3 Pruned Character *N*-grams

Like in Cavnar's method, we used pruned *n*-grams models of the reference or training as well as test data. For each language-encoding pair, some training data is provided. A character based *n*-gram model is prepared from this data. *N*-grams of all orders are combined and ranked according to frequency. A certain number of them (say 1000) with highest frequencies are retained and the rest are dropped. This gives us the pruned character *n*-grams model, which is used for language-encoding identification.

As an attempt to increase the performance, we also tried to augment the pruned character *n*-grams model with a word *n*-gram model.

4 Distance Measures

Some of the measures we have experimented with have already been mentioned in the section on previous work. The measures considered in this work range from something as simple as log probability difference to the one based on Jiang and Conrath (Jiang and Conrath, 1997) measure.

Assuming that we have two models or distributions *P* and *Q* over a variable *X*, the measures (*sim*) are defined as below (*p* and *q* being probabilities and *r* and *s* being ranks in models *P* and *Q*:

1. Log probability difference:

$$sim = \sum_x (log\, p(x) - log\, q(x)) \tag{1}$$

2. Absolute log probability difference:

$$sim = \sum_x (abs(log\, p(x)) - abs(log\, q(x))) \tag{2}$$

3. Cross entropy:

$$sim = \sum_x (p(x) * log\ q(x)) \tag{3}$$

4. RE measure (based on relative entropy or Kullback-Leibler distance – see note below):

$$sim = \sum_x p(x) \frac{log\ p(x)}{log\ q(x)} \tag{4}$$

5. JC measure (based on Jiang and Conrath's measure) (Jiang and Conrath, 1997):

$$sim = A - B \tag{5}$$

where,

$$A = 2 * \sum_x (log\ p(x) + log\ q(x)) \tag{6}$$

and,

$$B = \sum_x log\ p(x) + \sum_x log\ q(x) \tag{7}$$

6. Out of rank measure (Cavnar and Trenkle, 1994):

$$sim = \sum_x abs(r(x) - s(x)) \tag{8}$$

7. MRE measure (based on mutual or symmetric relative entropy, the original definition of KL-distance given by Kullback and Leibler):

$$sim = \sum_x p(x) \frac{log\ p(x)}{log\ q(x)} + \sum_x q(x) \frac{log\ q(x)}{log\ p(x)} \tag{9}$$

8. Mutual (or symmetric) cross entropy:

$$sim = \sum_x (p(x)*log\ q(x) + q(x)*log\ p(x)) \tag{10}$$

As can be noticed, all these measures, in a way, seem to be information theoretic in nature. However, our focus in this work is more on the presenting empirical evidence rather than discussing mathematical foundation of these measures. The latter will of course be interesting to look into.

NOTE:

We had initiallly experimented with relative entropy or KL-distance as defined below (instead of the RE measure mentioned above):

$$sim = \sum_x p(x)\ log \frac{p(x)}{q(x)} \tag{11}$$

Another measure we tried was DL measure (based on Dekang Lin's measure, on which the JC measure is based):

$$sim = \frac{A}{B} \tag{12}$$

where A and B are as given above.

The results for the latter measure were not very good (below 50% in all cases) and the RE measure defined above performed better than relative entropy. These results have not been reported in this paper.

5 Mutual Cross Entropy

Cross entropy is a well known distance measure used for various problems. *Mutual cross entropy* can be seen as bidirectional or symmetric cross entropy. It is defined simply as the sum of the cross entropies of two distributions with each other.

Our motivation for using 'mutual' cross entropy was that many similarity measures like cross entropy and relative entropy measure how similar one distribution is to the other. This will not necessary mean the same thing as measuring how similar two distributions are to each other. Mutual information measures this bidirectional similarity, but it needs joint probabilities, which means that it can only be applied to measure similarity of terms within one distribution. Relative entropy or Kullback-Leibler measure is applicable, but as the results show, it doesn't work as well as expected.

Note that some authors treat relative entropy and mutual information interchangeably. They are very similar in nature except that one is applicable for one variable in two distributions and the other for two variables in one distribution.

Our guess was that symmetric measures may give better results as both the models give some information about each other. This seems to be supported by the results for cross entropy, but (asymmetric) cross entropy and RE measures also gave good results.

6 The Algorithm

The foundation of the algorithm for identifying the language and encoding of a text or string has already been explained earlier. Here we give a summary of the algorithm we have used. The parameters for the algorithm and their values used in our experiments reported here have also been listed. These parameters allow the algorithm to be tuned

Table 1: DESCRIPTION OF DATA SETS

	Names	Total Count
Languages	Afrikaans (1), Assamese (1), Bengali (2), Bulgarian (1), Catalan (1) Czech (1), Danish (1), Dutch (1), English (1), Esperanto (1) Finnish (1), French (1), German (1), Gujarati (2), Hindi (8) Icelandic (1), Iloko (1), Iroquoian (1), Italian (1), Kannada (1) Khasi (1), Latin (1), Malayalam (1), Marathi (5), Modern Greek (1) Nahuatl (1), Norwegian (1), Oriya (2), Polish (1), Portugues (1) Punjabi (1), Romanian (1), Russian (1), Serbian (1), Spanish (1) Tagalog (1), Tamil (1), Telugu (1), Welsh (1)	39
Encodings	UTF8 (7), ISO-8859-1 (16), ISO-8859-2 (1), US-ASCII (4) Windows-1251 (2), Windows-1250 (1), ISCII (10), ISFOCB (1) ITrans (1), Shusha (1), Typewriter (1), WX (1), Gopika (1) Govinda (1), Manjusha (1), Saamanaa (1), Subak (1) Akruti Sarala (1), Webdunia (1)	19

Counts in parenthesis represent the extra ambiguity for that language or encoding.
For example, Hindi (8) means that 8 different encodings were tested for Hindi.

Language-Encoding Pairs: 53

Minimum training data size: 16035 characters (2495 words)

Maximum training data size: 650292 characters (102377 words)

Average training data size: 166198 characters (22643 words)

Confusable Languages: Assamese/Bengali/Oriya, Dutch/Afrikaans, Norwegian/Danish, Spanish/Tagalog, Hindi/Marathi, Telugu/Kannada/Malayalam, Latin/Franch

Table 2: NUMBER OF TEST SETS

Size	Number
100	22083
200	10819
500	4091
1000	1867
2000	1524
All test data	840

or customized for best performance. Perhaps they can even be learned by using some approach as the EM algorithm.

1. Train the system by preparing character based and word based (optional) n-grams from the training data.

2. Combine n-grams of all orders (O_c for characters and O_w for words).

3. Sort them by rank.

4. Prune by selecting only the top N_c character n-grams and N_w word n-grams for each language-encoding pair.

5. For the given test data or string, calculate the character n-gram based score sim_c with every model for which the system has been trained.

6. Select the t most likely language-encoding pairs (training models) based on this character based n-gram score.

7. For each of the t best training models, calculate the score with the test model. The score is calculated as:

$$score = sim_c + a * sim_w \qquad (13)$$

   where c and w represent character based and word based n-grams, respectively. And a is the weight given to the word based n-grams. In our experiment, this weight was 1 for the case when word n-grams were considered and 0 when they were not.

8. Select the most likely language-encoding pair out of the t ambiguous pairs, based on the combined score obtained from word and character based models.

Table 3: PRECISION FOR VARIOUS MEASURES AND TEST SIZES

Test Size (characters)		Precision							
		LPD	ALPD	CE	RE	CT	JC	MRE	MCE
100	CN	91.00	90.69	96.13	98.51	78.92	97.71	98.26	97.64
	CWN	94.31	94.15	97.50	75.54	81.63	98.35	94.16	98.38
200	CN	94.46	94.37	97.72	99.35	91.24	99.05	99.24	99.05
	CWN	96.52	96.52	98.85	90.54	92.79	99.21	91.13	99.39
500	CN	96.24	96.24	98.39	99.68	96.41	99.58	99.63	99.63
	CWN	98.19	97.80	99.46	94.65	96.82	99.63	98.78	99.85
1000	CN	97.18	96.81	98.81	99.78	97.73	99.89	99.73	99.95
	CWN	98.21	98.21	99.68	96.64	98.05	99.89	99.40	100.00
2000	CN	95.01	94.21	98.20	99.40	95.21	99.33	99.20	99.47
	CWN	96.74	97.14	99.47	94.01	95.81	99.40	96.67	99.60
All available test data	CN	82.50	88.57	98.33	99.88	94.76	99.88	99.76	100.00
	CWN	89.88	94.64	99.88	94.76	96.55	99.88	97.86	100.00

CN: Character n-grams only, *CWN:* Character n-grams plus word n-grams

To summarize, the parameters in the above method are:

1. Character based n-gram models P_c and Q_c
2. Word based n-gram models P_w and Q_w
3. Orders O_c and O_w of n-grams models
4. Number of retained top n-grams N_c and N_w (pruning ranks for character based and word based n-grams, respectively)
5. Number t of character based models to be disambiguated by word based models
6. Weight a of word based models

Parameters 3 to 6 can be used to tune the performace of the identification system. The results reported in this paper used the following values of these parameters:

1. $O_c = 4$
2. $O_w = 3$
3. $N_c = 1000$
4. $N_w = 500$
5. $t = 5$
6. $a = 1$

There is, of course, the type of similarity score, which can also be used to tune the performance. Since **MCE** gave the best overall performance in our experiments, we have selected it as the default score type.

7 Implementation

The language and encoding tool has been implemented as a small API in Java. This API uses another API to prepare pruned character and word n-grams which was developed as part of another project. A graphical user interface (GUI) has also been implemented for identifying the languages and encodings of texts, files, or batches of files. The GUI also allows a user to easily train the tool for a new language-encoding pair. The tool will be modified to work in client-server mode for documents from the Internet.

From implementation point of view, there are some issues which can significantly affect the performance of the system:

1. Whether the data should be read as text or as a binary file.

2. The assumed encoding used for reading the text, both for training and testing. For example, if we read UTF8 data as ISO-8859-1, there will be errors.

3. Whether the tranining models should be read every time they are needed or be kept in memory.

4. If training models are stored (even if they are only read at the beginning and then kept in memory), as will have to be done for practical applications, how should they be stored: as text or in binary files?

To take care of these issues, we adopted the following policy:

1. For preparing character based models, we read the data as binary files and the characters are read as bytes and stored as numbers. For word based models, the data is read as text and the encoding is assumed to be UTF8. This can cause errors, but it seems to be the best (easy) option as we don't know the actual encoding. A slightly more difficult option to implement would be to use character based models to guess the encoding and then build word based models using that as the assumed encoding. The problem with this method will be that no programming environment supports all possible encodings. Note that since we are reading the text as bytes rather than characters for preparing 'character based *n*-grams', technically we should say that we are using byte based *n*-grams models, but since we have not tested on multi-byte encodings, a byte in our experiments was almost always a character, except when the encoding was UTF8 and the byte represented some meta-data like the script code. So, for practical purposes, we can say that we are using character based *n*-grams.

2. Since after pruning, the size of the models (character as well as word) is of the order of 50K, we can afford to keep the training models in memory rather than reading them every time we have to identify the language and encoding of some data. This option is naturally faster. However, for some applications where language and encoding identification is to be done rarely or where there is a memory constraint, the other option can be used.

3. It seems to be better to store the training models in binary format since we don't know the actual encoding and the assumed encoding for storing may be wrong. We tried both options and the results were worse when we stored the models as text.

Our identification tool provides customizability with respect to all the parameters mentioned in this and the previous section.

8 Evaluation

Evaluation was performed for all the measures listed earlier. These are repeated here with a code for easy reference in table-3.

- **LPD**: Log probability difference
- **ALPD**: Absolute log probability difference
- **CE**: Cross entropy
- **RE**: RE measure based on relative entropy
- **JC**: JC measure (based on Jiang and Conrath's measure)
- **CT**: Cavnar and Trenkle's *out of rank* measure
- **MRE**: MRE measure based on mutual (symmetric) relative entropy
- **MCE**: Mutual (symmetric) cross entropy

We tested on six different sizes in terms of characters, namely 100, 200, 500, 1000, 2000, and all the available test data (which was not equal for various language-encoding pairs). The number of language-encoding pairs was 53 and the minimum number of test data sets was 840 when we used all available test data. In other cases, the number was naturally larger as the test files were split in fragments (see table-2).

The languages considered ranged from Esperanto and Modern Greek to Hindi and Telugu. For Indian languages, especially Hindi, several encodings were tested. Some of the pairs had UTF8 as the encoding, but the information from UTF8 byte format was not explicitly used for identification. The number of languages tested was 39 and number encodings was 19. Total number of language-encoding pairs was 53 (see table-1).

The test and training data for about half of the pairs was collected from web pages (such as Gutenberg). For Indian languages, most (but not all) data was from what is known as the CIIL corpus.

We didn't test on various training data sizes. The size of the training data ranged from 2495 to 102377 words, with more on the lower side than on the higher.

Note that we have considered the case where both the language and the encoding are unknown, not where one of them is known. In the latter case, the performance can only improve. Another point worth mentioning is that the training data was not very clean, i.e., it had noise (such as words or sentences from other languages). Error details have been given in table-4.

Table 4: ERROR DETAILS

Language-Encoding	**Identified As**
Afrikaans::ISO-8859-1	Dutch::ISO-8859-1 (9)
Assamese::ISCII	Bengali::ISCII (6), Oriya::ISCII (113)
Bengali::ISCII	Hindi::ISCII (2), Oriya::ISCII (193)
Bulgarian::Windows-1251	Marathi::ISCII (6)
Catalan::ISO-8859-1	Latin::ISO-8859-1 (4)
Danish::ISO-8859-1	Norwegian::ISO-8859-1 (7)
Dutch::ISO-8859-1	Afrikaans::ISO-8859-1 (4)
English::ASCII	Icelandic::UTF8 (36)
Esperanto::UTF8	Danish::ISO-8859-1 (5), Italian::ISO-8859-1 (1)
French::ISO-8859-1	Catalan::ISO-8859-1 (6)
German::ISO-8859-1	Dutch::ISO-8859-1 (4), Latin::ISO-8859-1 (3)
Hindi::ISCII	English::ASCII (14), Marathi::ISCII (20)
Hindi::Isfocb	Dutch::ISO-8859-1 (4), English::ASCII (6)
Hindi::Phonetic-Shusha	English::ASCII (14)
Hindi::Typewriter	English::ASCII (12)
Hindi::UTF8	Marathi::UTF8 (82)
Hindi::WX	English::ASCII (8)
Hindi::Webdunia	French::ISO-8859-1 (2), Gujarati::Gopika (9)
Icelandic::UTF8	Dutch::ISO-8859-1 (3), Latin::ISO-8859-1 (2)
Iloko::ISO-8859-1	Tagalog::ISO-8859-1 (18)
Iroquoian::ISO-8859-1	French::ISO-8859-1 (7)
Italian::ISO-8859-1	Catalan::ISO-8859-1 (2)
Kannada::ISCII	Malayalam::ISCII (9)
Latin::ISO-8859-1	Catalan::ISO-8859-1 (3), Dutch::ISO-8859-1 (85) French::ISO-8859-1 (28)
Malayalam::ISCII	Tamil::ISCII (3)
Marathi::ISCII	Hindi::ISCII (13)
Marathi::Manjusha	English::ASCII (1)
Marathi::UTF8	Hindi::UTF8 (30)
Nahuatl::ISO-8859-1	English::ASCII (2)
Norwegian::ISO-8859-1	Danish::ISO-8859-1 (69)
Oriya::ISCII	Assamese::ISCII (5), Bengali::ISCII (70), Hindi::ISCII (7)
Portugues::ISO-8859-1	Catalan::ISO-8859-1 (4)
Punjabi::ISCII	Assamese::ISCII (2), Hindi::ISCII (1)
Romanian::US-ASCII	Italian::ISO-8859-1 (2)
Russian::Windows-1251	Portugues::ISO-8859-1 (12)
Spanish::ISO-8859-1	Portugues::ISO-8859-1 (2), Tagalog::ISO-8859-1 (44)
Tagalog::ISO-8859-1	English::ASCII (37), Khasi::US-ASCII (15)
Telugu::ISCII	Hindi::ISCII (15), Kannada::ISCII (21), Malayalam::ISCII (2)

These error were for **MCE***, both with and without word models for all the test data sizes from 200 to all available data. Most of the errors were for smaller sizes, i.e., 100 and 200 characters.*

9 Results

The results are presented in table-3. As can be seen almost the measures gave at least moderately good results. The best results on the whole were obtained with mutual cross entropy. The JC measure gave almost equally good results. Even a simple measure like log probability difference gave surprisingly good results.

It can also be observed from table-3 that the size of the test data is an important factor in performance. More test data gives better results. But this does not always happen, which too is surprising. It means some other factors also come into play. One of these factors seem to whether the training data for different models is of equal size or not. Another factor seems to be noise in the data. This seems to affect some measures more than the others. For example, **LPD** gave the worst performance when all the available test data was used. For smaller data sets, noise is likely to get isolated in some data sets, and therefore is less likely to affect the results.

Using word *n*-grams to augment character *n*-grams improved the performance in most of the cases, but for measures like **JC**, **RE**, **MRE** and **MCE**, there wasn't much scope for improvement. In fact, for smaller sizes (100 and 200 characters), word models actually reduced the performance for these better measures. This means either that word models are not very good for better measures, or we have not used them in the best possible way, even though intuitively they seem to offer scope for improvement when character based models don't perform perfectly.

10 Issues and Enhancements

Although the method works very well even on little test and training data, there are still some issues and possible enhancements. One major issue is that Web pages quite often contain text in more than one language-encoding. An ideal language-encoding identification tool should be able to mark which parts of the page are in which language-encoding.

Another possible enhancement is that in the case of Web pages, we can also take into account the language and encoding specified in the Web page (HTML). Although it may not be correct for non-standard encodings, it might still be useful for differentiating between very close encodings like ASCII and ISO-8859-1 which might seem identical to our tool.

If the text happens to be in Unicode, then it might be possible to identify at least the encoding (the same encoding might be used for more than one languages, e.g., Devanagari for Hindi, Sanskrit and Marathi) without using a statistical method. This might be used for validating the result from the statistical method.

Since every method, even the best one, has some limitations, it is obvious that for practical applications we will have to combine several approaches in such a way that as much of the available information is used as possible and the various approaches complement each other. What is left out by one approach should be taken care of by some other approach. There will be some issues in combining various approaches like the order in which they have to used, their respective priorities and their interaction (one doesn't nullify the gains from another).

It will be interesting to apply the same method or its variations on text categorization or topic identification and other related problems. The distance measures can also be tried for other problems.

11 Conclusion

We have presented the results about some distance measures which can be applied to NLP problems. We also described a method for automatically identifying the language and encoding of a text using several measures including one called 'mutual cross entropy'. All these measures are applied on character based pruned *n*-grams models created from the training and the test data. There is one such model for each of the known language-encoding pairs. The character based models may be augmented with word based models, which increases the performance for not so good measures, but doesn't seem to have much effect for better measures. Our method gives good performance on a few words of test data and a few pages of training data for each language-encoding pair. Out of the measures considered, mutual cross entropy gave the best results, but **RE**, **MRE** and **JC** measures also performed almost equally well.

12 Acknowledgement

The author wishes to thank Preeti Pradhan, Nandini Upasani and Anita Chaturvedi of Language

Technologies Research Centre, International Institute of Information Technology, Hyderabad, India for helping in preparing the data for some of the language-encoding pairs. The comments of reviewers also helped in improving the paper.

References

Gary Adams and Philip Resnik. 1997. A language identification application built on the Java client-server platform. In Jill Burstein and Claudia Leacock, editors, *From Research to Commercial Applications: Making NLP Work in Practice*, pages 43–47. Association for Computational Linguistics.

K. Beesley. 1988. Language identifier: A computer program for automatic natural-language identification on on-line text.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

H. Combrinck and E. Botha. 1994. Automatic language identification: Performance vs. complexity. In *Proceedings of the Sixth Annual South Africa Workshop on Pattern Recognition.*

Ted Dunning. 1994. Statistical identification of language. Technical Report CRL MCCS-94-273, Computing Research Lab, New Mexico State University, March.

E. Giguet. 1995a. Categorization according to language: A step toward combining linguistic knowledge and statistic learning.

Emmanuel Giguet. 1995b. Multilingual sentence categorisation according to language. In *Proceedings of the European Chapter of the Association for Computational Linguistics, SIGDAT Workshop, From Text to Tags: Issues in Multilingual Language Analysis, Dublin, Ireland.*

Norman C. Ingle. 1976. A language identification table. In *The Incorporated Linguist, 15(4).*

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy.

G. Kikui. 1996. Identifying the coding system and language of on-line documents on the internet. In *COLING*, pages 652–657.

Bruno Martins and Mario J. Silva. 2005. Language identification in web pages. In *Proceedings of ACM-SAC-DE, the Document Engeneering Track of the 20th ACM Symposium on Applied Computing.*

Y. K. Muthusamy, E. Barnard, and R. A. Cole. 1994. Reviewing automatic language identification. In *IEEE Signal Processing Magazine*.

Patricia Newman. 1987. Foreign language identification - first step in the translation process. In *Proceedings of the 28th Annual Conference of the American Translators Association.*, pages 509–516.

Arjen Poutsma. 2001. Applying monte carlo techniques to language identification. In *Proceedings of CLIN.*

P. Sibun and J. C. Reynar. 1996. Language identification: Examining the issues. In *In Proceedings of SDAIR-96, the 5th Symposium on Document Analysis and Information Retrieval.*, pages 125–135.

Kranig Simon. 2005. Evaluation of language identification methods. In *BA Thesis*. Universitt Tbingens.

C. Souter, G. Churcher, J. Hayes, J. Hughes, and S. Johnson. 1994. Natural language identification using corpus-based models. In *Hermes Journal of Linguistics.*, pages 183–203.

Johnson Stephen. 1993. Solving the problem of language recognition. In *Technical Report*. School of Computer Studies, University of Leeds.

W. J. Teahan and D. J. Harper. 2001. Using compression based language models for text categorization. In *J. Callan, B. Croft and J. Lafferty (eds.), Workshop on Language Modeling and Information Retrieval.*, pages 83–88. ARDA, Carnegie Mellon University.